Mapping Bengaluru's Culinary Landscape: A Spatial Analysis of Zomato Restaurant Data

Author: Steve Githinji

Overview

The project aims to perform spatial analysis on restaurant data from Zomato in Bengaluru, India. Bengaluru is a bustling city known for its diverse culinary scene, and understanding the spatial distribution of restaurants and their attributes can provide valuable insights for various stakeholders, including Zomato, restaurant owners, and food enthusiasts.

Business Understanding

Zomato is a popular online restaurant discovery and food delivery platform that operates in many countries around the world. It allows users to search for restaurants, read reviews and ratings, view menus, and place orders for food delivery or pickup. Zomato also provides a variety of information about restaurants, including their location, operating hours, and contact details. It has become a widely used platform for finding and ordering food from a diverse range of restaurants, making it convenient for users to explore different cuisines and dining options. The key business understanding includes:

- **Geographic Distribution:** Understanding where restaurants are concentrated and identifying areas with a high density of dining options.
- **Customer Preferences:** Analyzing restaurant ratings to identify areas with a high concentration of highly-rated restaurants, providing insights into customer preferences and potential food hubs.
- **Market Insights:** Gaining insights into the competition among restaurants in different areas and identifying potential gaps in the market.
- **Operational Efficiency:** Exploring opportunities to optimize food delivery logistics and improve service quality based on restaurant locations.

Objectives

1. **Spatial Visualization:** Create visually appealing maps that displays the locations of all restaurants in Bengaluru. These maps will include three main components:
    - **Marker Points:** Display each restaurant as a marker point on the map, allowing users to click on individual markers for more information about each restaurant.
    - **Marker Clusters:** Group nearby restaurants into clusters to avoid clutter on the map, enhancing user experience.
    - **Heatmap:** Generate a heatmap layer that illustrates the overall density of restaurants across Bengaluru, with areas of higher density showing up as more intense colors on the map.
2. **High-Rating Restaurant Heatmap:** Create a separate heatmap layer that highlights areas in Bengaluru where highly-rated restaurants are concentrated. This will provide insights into the geographic distribution of top-rated dining options.
3. **Insightful Reporting:** Summarize key findings and insights gained from the spatial analysis. This report can be shared with Zomato to help them make informed decisions about marketing, partnerships, and service improvements.
4. **Recommendations:** Based on the analysis, provide recommendations for Zomato, restaurant owners, and other stakeholders. These recommendations may include targeting specific areas for marketing campaigns, identifying potential collaboration opportunities, or optimizing food delivery routes.

By achieving these objectives, this project will provide a comprehensive view of the restaurant landscape in Bengaluru, enabling stakeholders to make data-driven decisions that enhance the dining experience and the food industry in the city.

Data Understanding

We begin by importing relevant packages|

In [1]:
```
# Importing relevant packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from geopy.geocoders import Nominatim
import folium
from folium.plugins import HeatMap
from folium.plugins import FastMarkerCluster
from folium import Marker

from warnings import filterwarnings
filterwarnings('ignore')
```

Next we import the dataset.

In [2]:
```
# Import dataset
data = pd.read_csv(r'Data/zomato.csv')

# Inspect data type
type(data)
```

Out[2]:
```
pandas.core.frame.DataFrame
```

The imported data belongs to dataframe data-structure. We then inspect the first few rows of the data.

In [3]:
```
# Getting first 3 rows of data
data.head(3)
```

Out[3]:

	url	address	name	online_order	book_table	rate	votes	phone	location
0	https://www.zomato.com/bangalore/jalsa-banasha...	942, 21st Main Road, 2nd Stage, Banashankari, ...	Jalsa	Yes	Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari
1	https://www.zomato.com/bangalore/spice-elephan...	2nd Floor, 80 Feet Road, Near Big Bazaar, 6th ...	Spice Elephant	Yes	No	4.1/5	787	080 41714161	Banashankari
2	https://www.zomato.com/SanchurroBangalore?cont...	1112, Next to KIMS Medical College, 17th Cross...	San Churro Cafe	Yes	No	3.8/5	918	+91 9663487993	Banashankari

In [4]:
```
# Getting all the columns of dataframe
data.columns
```

Out[4]:
```
Index(['url', 'address', 'name', 'online_order', 'book_table', 'rate', 'votes',
       'phone', 'location', 'rest_type', 'dish_liked', 'cuisines',
       'approx_cost(for two people)', 'reviews_list', 'menu_item',
       'listed_in(type)', 'listed_in(city)'],
      dtype='object')
```

In [5]:
```
# Getting dimensions of dataframe
data.shape
```

Out[5]:
```
(51717, 17)
```

The data contains 51,717 rows and 17 columns.

Data Preprocessing

To begin data preprocessing, we first check for duplicates.

In [6]:
```
# Check for duplicates
data.duplicated().sum()
```

Out[6]:
```
0
```

The dataset does not contain any duplicates. We then check for missing values.

In [7]:
```
# Check for missing values
data.isnull().sum()
```

Out[7]:
```
url                            0
address                        0
name                           0
online_order                   0
book_table                     0
rate                        7775
votes                          0
phone                       1208
location                      21
rest_type                    227
dish_liked                 28078
cuisines                      45
approx_cost(for two people)  346
reviews_list                   0
menu_item                      0
listed_in(type)                0
listed_in(city)                0
dtype: int64
```

There are columns with missing values that need to be cleaned. We drop all rows with missing values in the `location` column.

In [8]:
```
# Drop rows with missing values in location column
data.dropna(subset=['location'], inplace=True)

# Confirm rows were dropped
data.isnull().sum()
```

Out[8]:
```
url                            0
address                        0
name                           0
online_order                   0
book_table                     0
rate                        7754
votes                          0
phone                       1187
location                       0
rest_type                    206
dish_liked                 28057
cuisines                      24
approx_cost(for two people)  325
reviews_list                   0
menu_item                      0
listed_in(type)                0
listed_in(city)                0
dtype: int64
```

We create a copy of our dataframe called `df` which will be used for our analysis.

In [9]:
```
# Create copy of fataframe
df = data.copy()
```

We then inspect the `location` column.

In [10]:
```
# View location column
df['location']
```

Out[10]:
```
0                    Banashankari
1                    Banashankari
2                    Banashankari
3                    Banashankari
4                    Basavanagudi
                     ...
51712                   Whitefield
51713                   Whitefield
51714                   Whitefield
51715    ITPL Main Road, Whitefield
51716    ITPL Main Road, Whitefield
Name: location, Length: 51696, dtype: object
```

The location data does not contain city, state and country. We engineer these features into the column to get more accurate geographical coordinates.

In [11]:
```
# Add city, state and country
df['location'] = df['location'] + ' , Bangalore , Karnataka , India'

# Preview location column
df['location']
```

Out[11]:
```
0             Banashankari , Bangalore , Karnataka , India
1             Banashankari , Bangalore , Karnataka , India
2             Banashankari , Bangalore , Karnataka , India
3             Banashankari , Bangalore , Karnataka , India
4             Basavanagudi , Bangalore , Karnataka , India
                              ...
51712           Whitefield , Bangalore , Karnataka , India
51713           Whitefield , Bangalore , Karnataka , India
51714           Whitefield , Bangalore , Karnataka , India
51715    ITPL Main Road, Whitefield , Bangalore , Karna...
51716    ITPL Main Road, Whitefield , Bangalore , Karna...
Name: location, Length: 51696, dtype: object
```

We look at the data types of all the features.

In [12]:
```
# Inspect data types
df.dtypes
```

Out[12]:
```
url                            object
address                        object
name                           object
online_order                   object
book_table                     object
rate                           object
votes                           int64
phone                          object
location                       object
rest_type                      object
dish_liked                     object
cuisines                       object
approx_cost(for two people)    object
reviews_list                   object
menu_item                      object
listed_in(type)                object
listed_in(city)                object
dtype: object
```

Extract Latitudes and Longitudes

The dataset does not contain geographical coordinates which are needed to plot our maps. We create a new dataframe with a column of all the locations called 'Name"

In [13]:
```
# New dataframe
rest_loc = pd.DataFrame()

# Add column with all locations
rest_loc['Name'] = df['location'].unique()

# Inspect new dataframe
rest_loc
```

Out[13]:

	Name
0	Banashankari , Bangalore , Karnataka , India
1	Basavanagudi , Bangalore , Karnataka , India
2	Mysore Road , Bangalore , Karnataka , India
3	Jayanagar , Bangalore , Karnataka , India
4	Kumaraswamy Layout , Bangalore , Karnataka , I...
...	...
88	West Bangalore , Bangalore , Karnataka , India
89	Magadi Road , Bangalore , Karnataka , India
90	Yelahanka , Bangalore , Karnataka , India
91	Sahakara Nagar , Bangalore , Karnataka , India
92	Peenya , Bangalore , Karnataka , India

93 rows × 1 columns

Using the geopy library's Nominatim geocoder to obtain latitude and longitude coordinates for restaurant locations based on their names and loop through our DataFrame of restaurant names and then retrieve the coordinates using geocoding.

In [14]:
```
# Create a geolocator object with a specified user agent and no timeout.
geolocator = Nominatim(user_agent='app1', timeout=None)

# Initialize empty lists to store latitude and longitude values.
lat = []
lon = []

# Loop iterates through the restaurant names
for name in rest_loc['Name']:
    """
    Retrieve location information based on name. If restaurant name not
    found in geocoding service, append NaN. If a valid location is found,
    append the latitude and longitude values to the respective lists.
    """
    location = geolocator.geocode(name)
    if location is None:
        lat.append(np.nan)
        lon.append(np.nan)
    else:
        lat.append(location.latitude)
        lon.append(location.longitude)

# Print latitude values for the restaurant locations.
print(lat)
```

```
[12.9152208, 12.9417261, 12.9467026, 12.9292731, 12.9081487, 12.9274413, 12.9658625, 12.9055682, 12.9151486, 1
2.9287596, 12.965717999999999, 12.9841958, 12.8769331, 12.911275849999999, 12.8683735, 12.9089453, 12.9856596,
12.848759900000001, 12.9116225, 12.9552572, 12.9237639, 12.9489339, 12.9575547, 12.9348429, 12.9408685, 12.970
0474, 12.9364846, 13.0464531, 12.9327778, 12.93103185, 12.9696365, 12.9892546, 12.9606699, 12.9732913, 12.9277
245, 12.9986827, 13.0227204, 12.9755264, 12.9736132, 12.9749487, 12.9742939, 12.9778793, 12.9741926, 12.98639
1, 12.9829856, 12.9744255, 12.987043, 12.983903, 12.9822323, 12.988721250000001, 13.0358698, 12.9624669, 12.94
5245, 12.9678074, 12.9835954, 13.0027353, 12.9931876, 13.0093455, 12.9390255, 12.978129800000001, 12.957998, 1
2.9746886, 12.9578658, 12.9668213, 12.9862452, 12.9413238, 13.007516, 12.9243692, 12.9282918, 12.9340114, 12.9
291385, 12.9882338, 13.0141618, 13.022234699999998, 13.0431413, 13.0258087, 13.0221416, 13.0268145, 13.078474
3, nan, 12.973936, 12.9846713, 13.0382184, 12.9229728, 12.99359355, nan, 12.9932236, 13.02383, 13.022234699999
998, 12.975608, 13.1006982, 13.0621474, 13.0329419]
```

We incorporate geographic coordinates into our DataFrame for further spatial analysis and visualization. We do this by adding latitude and longitude information to our rest_loc DataFrame by creating two new columns named 'lat' and 'lon' and assigning the latitude and longitude lists (lat and lon) to those columns.

In [15]:
```
# Add Latitudes and Longitudes to DataFrame
rest_loc['lat'] = lat
rest_loc['lon'] = lon

# Preview DataFrame
rest_loc.head()
```

Out[15]:

	Name	lat	lon
0	Banashankari , Bangalore , Karnataka , India	12.915221	77.573598
1	Basavanagudi , Bangalore , Karnataka , India	12.941726	77.575502
2	Mysore Road , Bangalore , Karnataka , India	12.946703	77.530070
3	Jayanagar , Bangalore , Karnataka , India	12.929273	77.582423
4	Kumaraswamy Layout , Bangalore , Karnataka , I...	12.908149	77.555318

We check the DataFrame for latitudes and longitudes that were not extracted.

In [16]:
```
# Check for NaN values
rest_loc.isnull().sum()
```

Out[16]:
```
Name    0
lat     2
lon     2
dtype: int64
```

It seems coordinates for 2 locations were not obtained. Print these locations.

In [17]:
```
# Print rows with NaN values
rest_loc[rest_loc['lat'].isnull()]
```

Out[17]:

	Name	lat	lon
79	Rammurthy Nagar , Bangalore , Karnataka , India	NaN	NaN
85	Sadashiv Nagar , Bangalore , Karnataka , India	NaN	NaN

We Google search missing coordinates and input them manually.

In [18]:
```
# For Rammurthy Nagar, Bangalore 13.0163° N, 77.6785° E
rest_loc['lat'][79] = 13.0184435
rest_loc['lon'][79] = 77.6781215

# For Sadashiv Nagar, Bangalore 13.010316° N, 77.580569° E
rest_loc['lat'][85] = 13.010316
rest_loc['lon'][85] = 77.580569

# Check for NaN values
rest_loc.isnull().sum()
```

Out[18]:
```
Name    0
lat     0
lon     0
dtype: int64
```

The dataset is now ready for analysis.

Data Analysis

Restaurant Location Density

Where are most restaurants located in Bengalure City?

Using the value_counts() method on our DataFrame column named 'location' to count the occurrences of each unique location in our DataFrame (df). This is a useful step for obtaining a summary of how many restaurants are located in each unique location.

In [19]:
```
# Summary of how many restaurants are located in each unique location
restaurant_locations = df['location'].value_counts().reset_index()

# Print sumaary
restaurant_locations
```

Out[19]:

	index	location
0	BTM , Bangalore , Karnataka , India	5124
1	HSR , Bangalore , Karnataka , India	2523
2	Koramangala 5th Block , Bangalore , Karnataka ...	2504
3	JP Nagar , Bangalore , Karnataka , India	2235
4	Whitefield , Bangalore , Karnataka , India	2144
...	...	...
88	Yelahanka , Bangalore , Karnataka , India	6
89	West Bangalore , Bangalore , Karnataka , India	6
90	Jakkur , Bangalore , Karnataka , India	3
91	Rajarajeshwari Nagar , Bangalore , Karnataka ,...	2
92	Peenya , Bangalore , Karnataka , India	1

93 rows × 2 columns

We have 93 unique locations in our dataset.

We rename the columns from 'index' to 'Name' and from 'location' to 'Count' to make them more informative.

In [20]:
```
# Rename columns
restaurant_locations.columns = ['Name', 'Count']

# Preview dataframe
restaurant_locations.head()
```

Out[20]:

	Name	Count
0	BTM , Bangalore , Karnataka , India	5124
1	HSR , Bangalore , Karnataka , India	2523
2	Koramangala 5th Block , Bangalore , Karnataka ...	2504
3	JP Nagar , Bangalore , Karnataka , India	2235
4	Whitefield , Bangalore , Karnataka , India	2144

Merge 2 dataframes to get Name, 'Count', lat' and 'lot' using a common column 'Name'. We want to obtain a DataFrame of unique locations, their count, latitudes and longitudes.

In [25]:
```
# Merge DataFrames
beng_restaurant_locations = rest_loc.merge(restaurant_locations, on='Name')

# View DataFrame
beng_restaurant_locations
```

Out[25]:

	Name	lat	lon	Count
0	Banashankari , Bangalore , Karnataka , India	12.915221	77.573598	906
1	Basavanagudi , Bangalore , Karnataka , India	12.941726	77.575502	684
2	Mysore Road , Bangalore , Karnataka , India	12.946703	77.530070	22
3	Jayanagar , Bangalore , Karnataka , India	12.929273	77.582423	1926
4	Kumaraswamy Layout , Bangalore , Karnataka , I...	12.908149	77.555318	195
...	...	...	...	...
88	West Bangalore , Bangalore , Karnataka , India	13.022235	77.567183	6
89	Magadi Road , Bangalore , Karnataka , India	12.975608	77.555356	34
90	Yelahanka , Bangalore , Karnataka , India	13.100698	77.596345	6
91	Sahakara Nagar , Bangalore , Karnataka , India	13.062147	77.580061	53
92	Peenya , Bangalore , Karnataka , India	13.032942	77.527325	1

93 rows × 4 columns

Heatmap

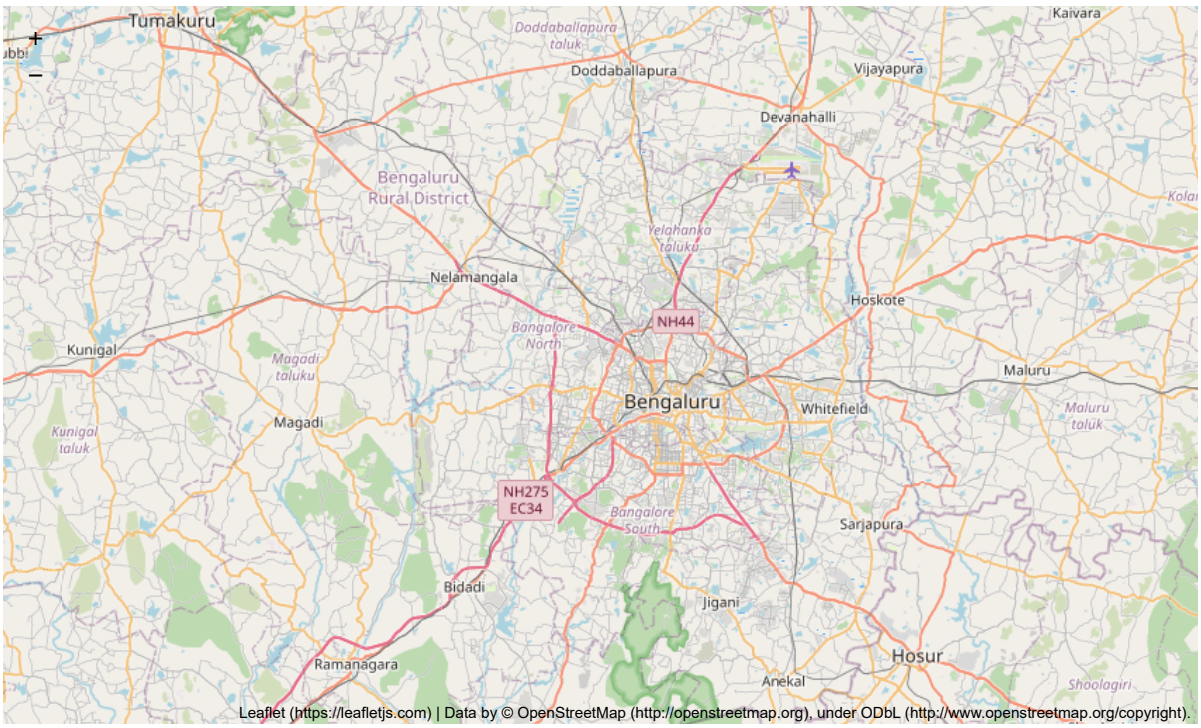
We define a function named generate_basemap that uses the Folium library to create a basemap centered around the coordinates [12.97, 77.59]. This function is set up to return the generated basemap.

In [22]:
```
# Function to generate a basemap
def generate_basemap():
    basemap = folium.Map(location=[12.97, 77.59])
    return basemap

# Use functtion to generate basemap
basemap = generate_basemap()

# Visualize map
basemap
```

Out[22]:



We use the Folium library to add a heatmap layer to the basemap. This code should result in an interactive Folium map that displays a heatmap of restaurant locations in Bengaluru, India, with the intensity of the heatmap corresponding to the count of restaurants in different areas.

In [23]:
```
# Create a heatmap layer and it to the basemap
HeatMap(beng_restaurant_locations[['lat', 'lon', 'Count']]).add_to(basemap)

# Display the updated basemap
basemap
```

Out[23]:

Marker Cluster Analysis

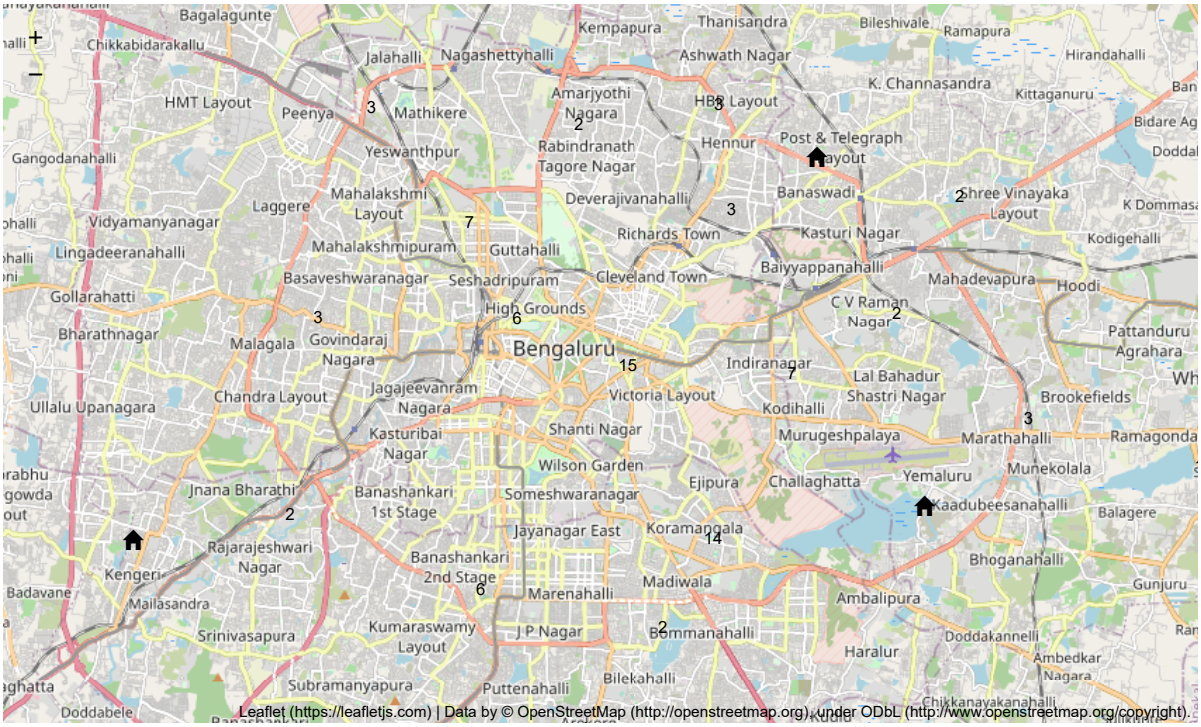
We create a new basemap named basemap2 using the generate_basemap function and then added a FastMarkerCluster layer to it. This will result in an interactive Folium map that displays clustered markers representing restaurant locations in Bengaluru, India. The clustering helps to avoid clutter on the map when there are many markers in close proximity.

In [24]:
```
# Generate a new basemap
basemap2 = generate_basemap()

# Create a FastMarkerCluster layer and add it to basemap
FastMarkerCluster(beng_restaurant_locations[['lat', 'lon', 'Count']]).add_to(basemap2)

# Display the updated map
basemap2
```

Out[24]:



Markers of all locations

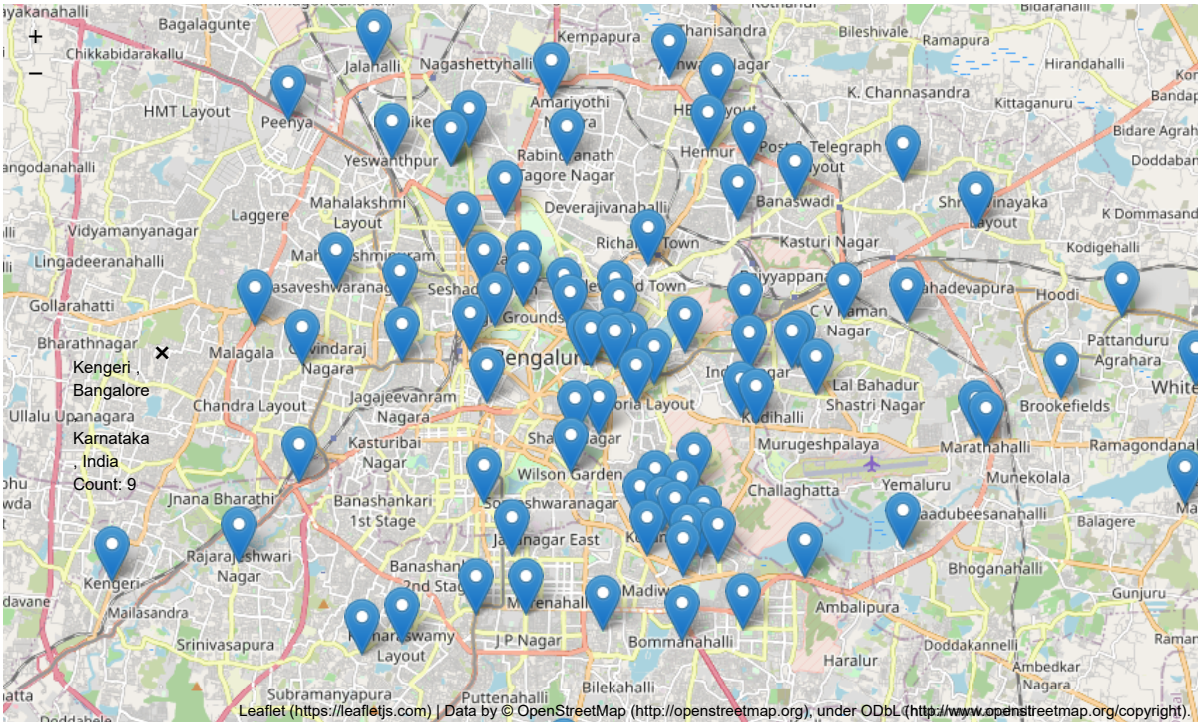
Create a new basemap named basemap3 using the generate_basemap function and then add individual marker points for restaurant locations. This will result in an interactive Folium map that displays individual markers for each location in Bengaluru, India. When you click on a marker, a popup will appear, showing the location's name and count of restaurants. This allows users to explore the distribution of restaurants across the city and view additional information about each restaurant by clicking on the markers.

In [28]:
```
# Generate a new basemap
basemap3 = generate_basemap()

# Iterate through the rows
for idx, row in beng_restaurant_locations.iterrows():
    """Create an individual marker that includes a popup
    with the location name and count. Add each marker to
    the basemap.
    """
    Marker(location=[row['lat'], row['lon']], popup=f"{row['Name']} Count: {row['Count']}").add_to(basemap3)

# Display updated map
basemap3
```

Out[28]:



Location of Restaurants with high average rating

Inspect the unique restaurant rating values.

In [30]:
```
# Unique ratings
df['rate'].unique()
```

Out[30]:
```
array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
       '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
       '4.3/5', 'NEW', '2.9/5', '3.5/5', nan, '2.6/5', '3.8 /5', '3.4/5',
       '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
       '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
       '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
       '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
       '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
       '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
       '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)
```

There are some missing values and some values "NEW" and "-". The values also contain a suffix "/5" which should be removed.

What percentage of ratings are missing?

In [31]:
```
# Check % of missing values
df['rate'].isnull().sum()/len(df)*100
```

Out[31]:
```
14.999226245744351
```

Approximately 15% of rating values are missing. These rows will be dropped.

In [32]:
```
# Drop rows with missing values in rate column
df.dropna(subset=['rate'], inplace=True)

# Replace "NEW" and "-" with 0
df['rate'].replace('NEW', '0', inplace=True)
df['rate'].replace('-', '0', inplace=True)

# Remove suffix "/5"
df['rating'] = df['rate'].str.replace('/5', '')
df['rating'] = df['rating'].astype(float)

# Confirm updates
df['rating'].unique()
```

Out[32]:
```
array([4.1, 3.8, 3.7, 3.6, 4.6, 4. , 4.2, 3.9, 3.1, 3. , 3.2, 3.3, 2.8,
       4.4, 4.3, 0. , 2.9, 3.5, 2.6, 3.4, 4.5, 2.5, 2.7, 4.7, 2.4, 2.2,
       2.3, 4.8, 4.9, 2.1, 2. , 1.8])
```

The restaurants need to be grouped by location and the rating aggregated by mean and name by count.

In [33]:
```
# Group by location and aggregate to get new DataFrame
avg_rating_df = df.groupby(by=['location'], as_index=False).agg({'rating':'mean', 'name':'size'})

# Rename columns
avg_rating_df.columns = ['Name', 'Avg_rating', 'Count']

# View DataFrame
avg_rating_df
```

Out[33]:

	Name	Avg_rating	Count
0	BTM , Bangalore , Karnataka , India	3.296128	4261
1	Banashankari , Bangalore , Karnataka , India	3.373292	805
2	Banaswadi , Bangalore , Karnataka , India	3.362926	499
3	Bannerghatta Road , Bangalore , Karnataka , India	3.271677	1324
4	Basavanagudi , Bangalore , Karnataka , India	3.478185	628
...	...	...	...
87	West Bangalore , Bangalore , Karnataka , India	2.020000	5
88	Whitefield , Bangalore , Karnataka , India	3.384170	1693
89	Wilson Garden , Bangalore , Karnataka , India	3.257635	203
90	Yelahanka , Bangalore , Karnataka , India	3.640000	5
91	Yeshwantpur , Bangalore , Karnataka , India	3.502679	112

92 rows × 3 columns

Filtering our DataFrame to include only rows where the 'Count' column (representing the count of locations) is greater than 400.

In [34]:
```
# Filter to include Count>400
avg_rating_df = avg_rating_df[avg_rating_df['Count'] > 400]

# View updated DataFrame
avg_rating_df
```

Out[34]:

	Name	Avg_rating	Count
0	BTM , Bangalore , Karnataka , India	3.296128	4261
1	Banashankari , Bangalore , Karnataka , India	3.373292	805
2	Banaswadi , Bangalore , Karnataka , India	3.362926	499
3	Bannerghatta Road , Bangalore , Karnataka , India	3.271677	1324
4	Basavanagudi , Bangalore , Karnataka , India	3.478185	628
6	Bellandur , Bangalore , Karnataka , India	3.309833	1078
8	Brigade Road , Bangalore , Karnataka , India	3.595849	1084
9	Brookefield , Bangalore , Karnataka , India	3.374699	581
12	Church Street , Bangalore , Karnataka , India	3.963091	550
15	Cunningham Road , Bangalore , Karnataka , India	3.901053	475
16	Domlur , Bangalore , Karnataka , India	3.385548	429
19	Electronic City , Bangalore , Karnataka , India	3.041909	964
20	Frazer Town , Bangalore , Karnataka , India	3.564879	578
22	HSR , Bangalore , Karnataka , India	3.484070	2128
27	Indiranagar , Bangalore , Karnataka , India	3.652169	1936
29	JP Nagar , Bangalore , Karnataka , India	3.412926	1849
31	Jayanagar , Bangalore , Karnataka , India	3.615250	1718
35	Kalyan Nagar , Bangalore , Karnataka , India	3.529144	748
36	Kammanahalli , Bangalore , Karnataka , India	3.499810	525
40	Koramangala 1st Block , Bangalore , Karnataka ...	3.263938	965
43	Koramangala 4th Block , Bangalore , Karnataka ...	3.814352	864
44	Koramangala 5th Block , Bangalore , Karnataka ...	3.901512	2381
45	Koramangala 6th Block , Bangalore , Karnataka ...	3.662466	1111
46	Koramangala 7th Block , Bangalore , Karnataka ...	3.747842	1089
50	Lavelle Road , Bangalore , Karnataka , India	4.042886	499
51	MG Road , Bangalore , Karnataka , India	3.740550	836
54	Malleshwaram , Bangalore , Karnataka , India	3.668237	658
55	Marathahalli , Bangalore , Karnataka , India	3.400532	1503
59	New BEL Road , Bangalore , Karnataka , India	3.583174	523
66	Rajajinagar , Bangalore , Karnataka , India	3.422382	487
69	Residency Road , Bangalore , Karnataka , India	3.844572	608
70	Richmond Road , Bangalore , Karnataka , India	3.688013	634
75	Sarjapur Road , Bangalore , Karnataka , India	3.473558	919
82	Ulsoor , Bangalore , Karnataka , India	3.541398	901
88	Whitefield , Bangalore , Karnataka , India	3.384170	1693

We merge DataFrames with Name as the common column to include latitudes and longitudes.

In [35]:
```
# Merge DataFrames
ratings_locations = avg_rating_df.merge(rest_loc, on='Name')

# View merged DataFrame
ratings_locations
```

Out[35]:

	Name	Avg_rating	Count	lat	lon
0	BTM , Bangalore , Karnataka , India	3.296128	4261	12.911276	77.604565
1	Banashankari , Bangalore , Karnataka , India	3.373292	805	12.915221	77.573598
2	Banaswadi , Bangalore , Karnataka , India	3.362926	499	13.014162	77.651854
3	Bannerghatta Road , Bangalore , Karnataka , India	3.271677	1324	12.876933	77.595132
4	Basavanagudi , Bangalore , Karnataka , India	3.478185	628	12.941726	77.575502
5	Bellandur , Bangalore , Karnataka , India	3.309833	1078	12.931032	77.678247
6	Brigade Road , Bangalore , Karnataka , India	3.595849	1084	12.973613	77.607472
7	Brookefield , Bangalore , Karnataka , India	3.374699	581	12.966821	77.716889
8	Church Street , Bangalore , Karnataka , India	3.963091	550	12.974294	77.652519
9	Cunningham Road , Bangalore , Karnataka , India	3.901053	475	12.987043	77.594924
10	Domlur , Bangalore , Karnataka , India	3.385548	429	12.962467	77.638196
11	Electronic City , Bangalore , Karnataka , India	3.041909	964	12.848760	77.648253
12	Frazer Town , Bangalore , Karnataka , India	3.564879	578	12.998683	77.615525
13	HSR , Bangalore , Karnataka , India	3.484070	2128	12.911623	77.638862
14	Indiranagar , Bangalore , Karnataka , India	3.652169	1936	12.973291	77.640467
15	JP Nagar , Bangalore , Karnataka , India	3.412926	1849	12.915149	77.585702
16	Jayanagar , Bangalore , Karnataka , India	3.615250	1718	12.929273	77.582423
17	Kalyan Nagar , Bangalore , Karnataka , India	3.529144	748	13.022142	77.640337
18	Kammanahalli , Bangalore , Karnataka , India	3.499810	525	13.009346	77.637709
19	Koramangala 1st Block , Bangalore , Karnataka ...	3.263938	965	12.927725	77.632782
20	Koramangala 4th Block , Bangalore , Karnataka ...	3.814352	864	12.932778	77.629405
21	Koramangala 5th Block , Bangalore , Karnataka ...	3.901512	2381	12.934843	77.618977
22	Koramangala 6th Block , Bangalore , Karnataka ...	3.662466	1111	12.939025	77.623848
23	Koramangala 7th Block , Bangalore , Karnataka ...	3.747842	1089	12.936485	77.613478
24	Lavelle Road , Bangalore , Karnataka , India	4.042886	499	12.974949	77.599725
25	MG Road , Bangalore , Karnataka , India	3.740550	836	12.975526	77.606790
26	Malleshwaram , Bangalore , Karnataka , India	3.668237	658	13.002735	77.570325
27	Marathahalli , Bangalore , Karnataka , India	3.400532	1503	12.955257	77.698416
28	New BEL Road , Bangalore , Karnataka , India	3.583174	523	13.026815	77.571719
29	Rajajinagar , Bangalore , Karnataka , India	3.422382	487	12.988234	77.554883
30	Residency Road , Bangalore , Karnataka , India	3.844572	608	12.974193	77.611060
31	Richmond Road , Bangalore , Karnataka , India	3.688013	634	12.970047	77.617104
32	Sarjapur Road , Bangalore , Karnataka , India	3.473558	919	12.923764	77.654073
33	Ulsoor , Bangalore , Karnataka , India	3.541398	901	12.977879	77.624670
34	Whitefield , Bangalore , Karnataka , India	3.384170	1693	12.969637	77.749745

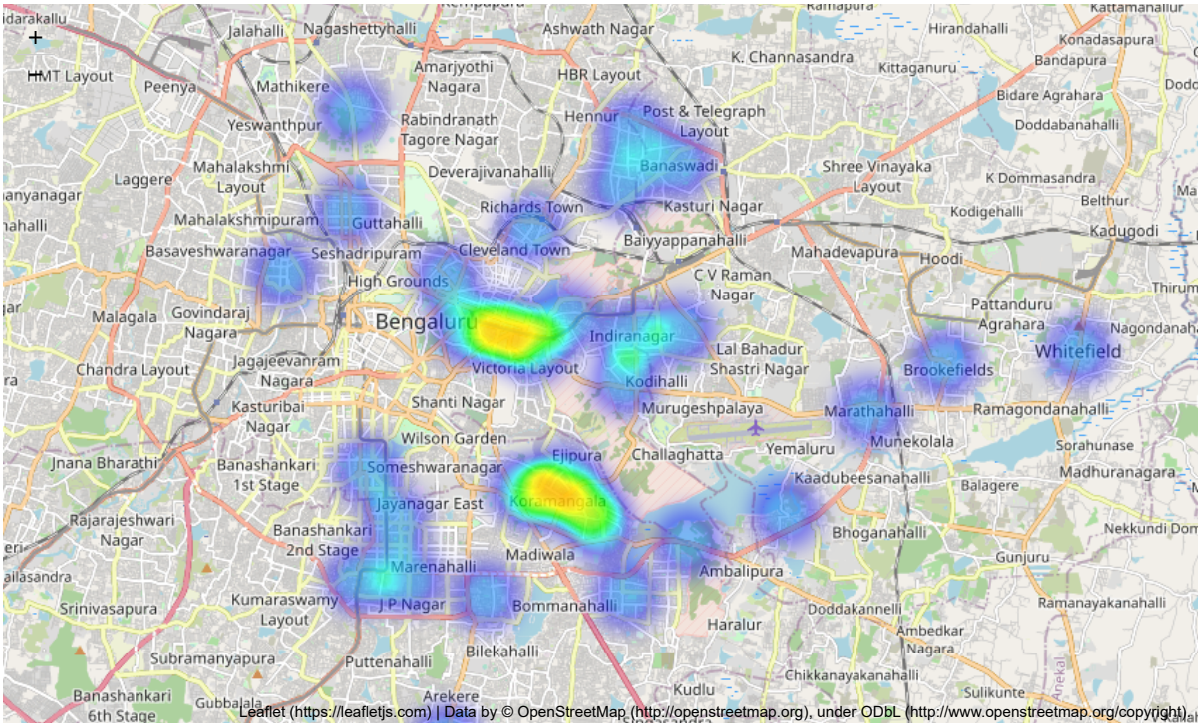
We plot an interactive Folium map that displays a heatmap of restaurant locations in Bengaluru, India, with the intensity of the heatmap corresponding to the average ratings of the restaurants at different locations.

In [36]:
```
# Generate a new basemap
basemap4 = generate_basemap()

# Create a heatmap layer, add heatmap layer to basemap4
HeatMap(ratings_locations[['lat', 'lon', 'Avg_rating']]).add_to(basemap4)

# Display updated basemap4
basemap4
```

Out[36]:



Conclusions

In this project, we conducted a comprehensive spatial analysis of restaurant data from Zomato in Bengaluru, India. Our analysis provided valuable insights into the culinary landscape of the city.

Restaurants with high ratings are primarily concentrated in the Koramangala area. This concentration suggests that Koramangala is a hotspot for highly-rated dining options, attracting food enthusiasts seeking quality experiences. The identification of areas with a high concentration of well-rated restaurants, such as Koramangala, presents opportunities for marketing campaigns and partnerships to promote these culinary hubs.

The Central Business District (CBD) of Bengaluru emerged as the epicenter of the restaurant scene. It hosts a significant portion of the city's dining establishments, reflecting its role as a hub for both business and leisure activities.

The spatial analysis can help food delivery platforms like Zomato optimize their delivery logistics, ensuring efficient and timely service to customers across the city. Understanding restaurant locations can lead to more effective delivery route planning.

Recommendations

From the analysis, we recommend the following:

- Collaborate with restaurants in Koramangala to enhance visibility and customer engagement, leveraging the concentration of highly-rated establishments.
- Explore ways to support restaurants in other emerging food districts to promote a diverse culinary scene.
- Continuously monitor and analyze customer reviews and preferences to adapt to evolving food trends and maintain a competitive edge.

In conclusion, this spatial analysis project offers a clear understanding of the restaurant landscape in Bengaluru, allowing stakeholders to make data-driven decisions that benefit both consumers and the food industry. The insights gained can inform strategies for marketing, partnerships, and service improvements, contributing to the vibrant food culture of the city.