Project by:
Bhakti Dhongade - UCE2022454 (SY Comp A3)
Ketaki Dixit - UCE2022457 (SY Comp A3)

Assignment 7

Problem Statement:

Develop a restaurant recommendation system based on user-specified cuisine preferences by analyzing restaurant descriptions. Design a system that aims to suggest similar restaurants to users, enhancing their dining experience and exploration of diverse cuisines.

Introduction:

In an era where dining experiences are becoming increasingly diverse, finding restaurants that match one's culinary preferences can be challenging. Traditional recommendation systems often rely on basic filters or user ratings, overlooking the nuanced descriptions that encapsulate a restaurant's unique offerings. This project introduces a novel approach leveraging content-based filtering and natural language processing techniques to analyze restaurant descriptions and recommend similar establishments based on user-input cuisine preferences. By harnessing TF-IDF vectorization and a sigmoid kernel, the system can capture semantic similarities between restaurant descriptions, enabling more personalized recommendations. This content-based filtering approach ensures that recommendations are based on the inherent characteristics of the restaurants themselves, rather than relying solely on user ratings or collaborative filtering methods.

Algorithm used: **Content-based Filtering**
Methods used:

1. **TF-IDF Vectorization:** It applies TF-IDF vectorization to the 'description' column of the DataFrame. TF-IDF stands for Term Frequency-Inverse Document Frequency, a numerical statistic that reflects how important a word is to a document in a collection or corpus.

2. **Computing Sigmoid Kernel:** It computes the sigmoid kernel matrix using the TF-IDF matrix. Sigmoid kernel is a similarity measure used to calculate the similarity between two samples.

Dataset Information:

Dataset Link: [Restaurant Dataset]

Sample Data:

```
print(df.iloc[0])
```

```
market                                                    Houston, TX
name                                                       McDonald's
description                                  $ • American, Burgers
displayAddress          2017 Main St Parking, 2017 Main St, Houston, T...
street                                                 2017 Main street
city                                                           Houston
state                                                               TX
zipCode                                                       77002.0
timezone                                                   US/Central
latitude                                                     29.748692
longitude                                                   -95.370802
averageRating                                                 3.850296
ratingCount                                                       5237
priceRange                                                           $
imageUrl                https://img.cdn4dd.com/cdn-cgi/image/fit=conta...
asapDeliveryAvailable                                             True
asapDeliveryTimeMinutes                                             17
pickupAvailable                                                  False
asapPickupAvailable                                              False
asapPickupMinutes                                                   10
weighted_average                                              3.833257
Name: 0, dtype: object
```

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
import folium
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import sigmoid_kernel
from folium import Map, Marker, Popup
from IPython.display import display

# Mount Google Drive
drive.mount('/content/drive')

# Load the dataset
df = pd.read_csv('/content/drive/MyDrive/new_clean_resto_csv.csv')

# Define TF-IDF Vectorizer
tfv = TfidfVectorizer(min_df=3, max_features=None,
```

```
                      strip_accents='unicode', analyzer='word',
token_pattern=r'\w{1,}',
                      ngram_range=(1, 3), stop_words='english')

# Fill NaNs with empty string
df['description'] = df['description'].fillna('')

# Fit the TF-IDF on the 'description' text
tfv_matrix = tfv.fit_transform(df['description'])

# Compute the sigmoid kernel
sig = sigmoid_kernel(tfv_matrix, tfv_matrix)
# print(type(sig))
# print(sig.shape)

# Reverse mapping of indices and restaurant names
indices = pd.Series(df.index, index=df['name']).drop_duplicates()

def give_rec(cuisine, sig=sig, num_recommendations=15):
    # Split the user input cuisine string
    cuisine_keywords = cuisine.split(',')

    # Filter restaurants based on cuisine
    cuisine_restaurants =
df[df['description'].str.lower().str.contains('|'.join(cuisine_keywords),
case=False)]
    # print(cuisine_restaurants.shape)
    # If no restaurants found with the specified cuisine, return None
    if cuisine_restaurants.empty:
        return None

    # Get the indices corresponding to the filtered restaurants
    idx = [indices[name] for name in cuisine_restaurants['name']]
    # print(type(idx))

    # Get the pairwise similarity scores
    sig_scores = [(i, sig[i]) for i in idx]

    # Sort the restaurants based on the average similarity score
```

```
    sig_scores = sorted(sig_scores, key=lambda x: np.mean(x[1]),
reverse=True)

    # Exclude the input restaurant itself
    sig_scores = sig_scores[1:num_recommendations+1]
    # print(type(sig_scores))
    # print(sig_scores.shape())

    # Get the indices of the recommended restaurants
    restaurant_indices = [i[0] for i in sig_scores]

    # Get the names of the recommended restaurants
    recommended_restaurants = df.iloc[restaurant_indices][['name']]
    # print(type(recommended_restaurants))
    # print(len(recommended_restaurants))

    return recommended_restaurants


# Take user input for cuisine
user_cuisine = input("Enter a cuisine: ")

# Get recommended restaurant names based on user input cuisine
recommended_restaurants = give_rec(user_cuisine)

# If no restaurants found with the specified cuisine, print a message
if recommended_restaurants is None:
    print("No restaurants found with the specified cuisine.")
else:
    # Create a map centered around the first recommended restaurant
    map_center = [df.loc[df['name'] ==
recommended_restaurants.iloc[0]['name'], 'latitude'].iloc[0],
                  df.loc[df['name'] ==
recommended_restaurants.iloc[0]['name'], 'longitude'].iloc[0]]
    mymap = folium.Map(location=map_center, zoom_start=12)

    # Add markers for each recommended restaurant
    for name in recommended_restaurants['name']:
        latitude = df.loc[df['name'] == name, 'latitude'].iloc[0]
        longitude = df.loc[df['name'] == name, 'longitude'].iloc[0]
```

```
        address = df.loc[df['name'] == name, 'displayAddress'].iloc[0]
        rating = df.loc[df['name'] == name, 'averageRating'].iloc[0]
        ratecount = df.loc[df['name'] == name, 'ratingCount'].iloc[0]
        delivery = df.loc[df['name'] == name,
'asapDeliveryAvailable'].iloc[0]
        pickup = df.loc[df['name'] == name, 'pickupAvailable'].iloc[0]
        url =
"https://www.zomato.com/bangalore/kirthis-biryani-banashankari-bangalore?c
ontext=eyJzZSI6eyJlIjpbIjE4MjAyOTk3IiwiNTkwOTAiLCIxODU1OTIxMSIsIjE4MzIzNjM
5IiwxODU3NDI3NywiMTg3MjM0OTQiLCIxODg2NzEyMiIsIjE4Njg4NDg5IiwiMTg1MjkwOTAiL
CI1NjE2NiIsIjYwOTE5IiwiMTgzOTY3NjAiLCIxODU5MzUxOSIsIjE4NTAwMDIwIiwiMTg4NDQ
yNjAiLCI1Nzk5NCIsIjU1MDkwIiwiNTc0MzUiXSwidCI6IkNhZlx1MDBlOXMgYW5kIERlbGlzI
GluIEJhbmFzaGFua2FyaSJ9fQ=="
        # Wrap the URL in an anchor tag
        url_html = f'<a href="{url}" target="_blank"><u>Go to Restaurant
Website</u></a>'
        description = "<b>Name:</b> " + name + "<br>" + "<b>Address:</b> "
+ address + "<br>" + "<b>Rating:</b> " + str(rating) + "<br>" + "<b>Rating
Count:</b> " + str(ratecount) + "<br>" + "<b>Delivery Available:</b> " +
str(delivery) + "<br>" + "<b>Pickup Available:</b> " + str(pickup) +
"<br>" + url_html

        # Create a Marker with a Popup containing HTML for full
description
        popup_content = f"""
            <!DOCTYPE html>
            <html>
            <head>
                <style>
                    .popup-text {{
                        max-height: 300px;
                    }}
                </style>
            </head>
            <body>
                <div class="popup-text">
                    {description}
                </div>
            </body>
            </html>
```

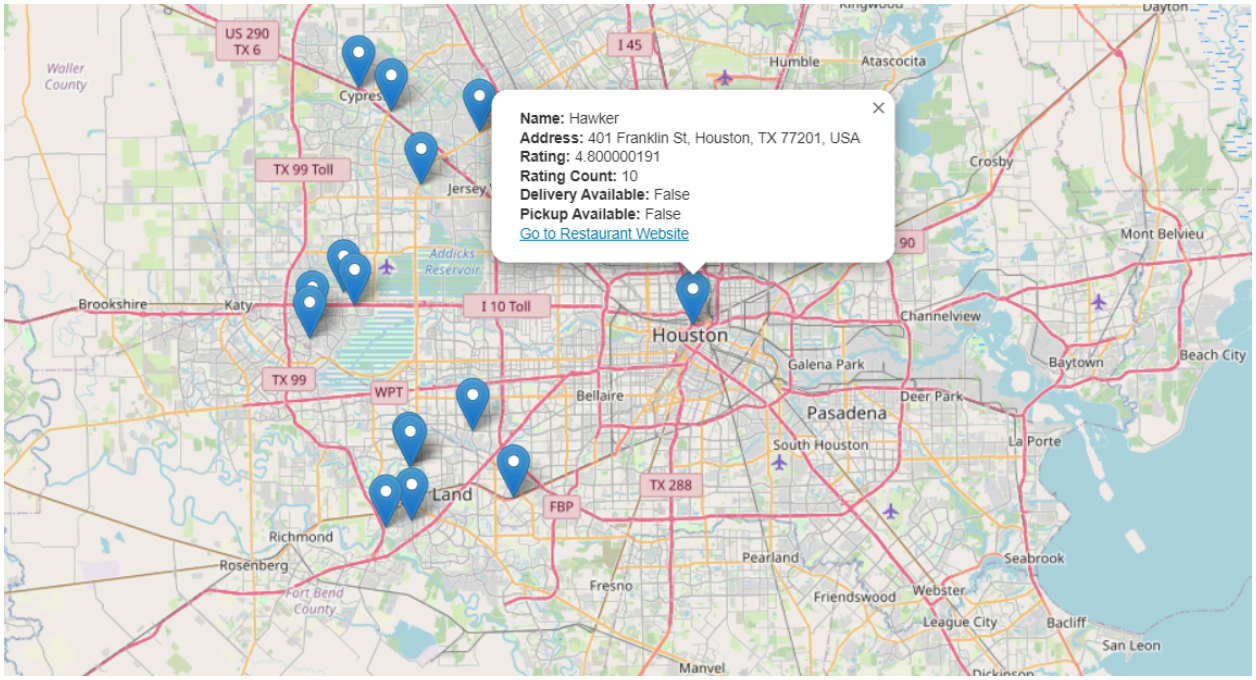
```
        """

        marker = Marker(location=[latitude, longitude], tooltip=name)
        popup = Popup(popup_content, max_width=300, max_height=300)  # Set
max_width and max_height for scrollable popup
        marker.add_child(popup)
        marker.add_to(mymap)

    print(recommended_restaurants)
    # Display the map
    display(mymap)
```

Output:

```
Enter a cuisine: Indian
                                name
1615                      Masala Wok
2181             TruIndia Restaurant
1750  Tandoori Hut Indian Restaurant
2494                   CZ Restaurant
2530                 Al Hamrah Grill
2558        Chettinad Indian Cuisine
2715                  Bombay Boo-yah
84                            Hawker
2005                    Cafe Yasmeen
1426                  Telfair Spices
1899                        Chat Hut
1971                      Curry Home
2176                     Kurry Walah
2447        Elite indopak Restaurant
1182             Monks Indian Bistro
```



Conclusion:

The restaurant recommendation system presented in this project offers a novel approach to assist users in discovering dining options tailored to their cuisine preferences. By leveraging natural language processing techniques and geographical visualization, the system enhances the user experience by providing personalized recommendations and facilitating exploration of diverse culinary offerings.

References:

GeeksforGeeks
TF-IDF
NLP