PREDICTING OPTIMAL CITY & LOCATION FOR A BUSINESS VENTURE

GROUP 15

GROUP MEMBERS:

1) **AKASH VASANTHAN**
2) **ALLAN JOSE SABU**
3) **MANVI LATHER**
4) **RIFHAD PIRANI**
5) **AKHIL KUMAR KOKKULA**

Table of Contents

INTRODUCTION

In this segment, a description of the project, the scope, background, and target audience are being discussed to answer what the project and the report tries to solve.

SCOPE

The scope of this project is limited to restaurants in New York, Los Angeles, and Chicago while the extensibility of the project is limitless. The project and the codes can be applied to any business ventures.

BACKGROUND

In this project we inspect three major cities in the United States (New York, Los Angeles, and Chicago) considering the population density, touristic destination aspect and number of hotels. One of the major interests that the people have when they visit a city for the first time is its people's eating habits. This study will be of interest for investors to make the right business decisions.

PROBLEM

The aim of this project is to find an optimal location for opening a restaurant from a pool of major cities in the United States. This project also tries to predict the ideal cuisine that is profitable. We consider the downtown location of each city and its coordinates for analysis. We have chosen the location for its foot traffic, its population density and number of hotels . The data is being extracted and the level of competition, trending cuisine is determined from the extracted Data.

INTEREST

This project can be very helpful for investors looking to start any business (restaurants in this project) in any city around the world (New York, Los Angeles, and Chicago in this project) and trying to follow the trend at a particular location. This project and the report provide insights on the hospitality and food industries around the city center of the above-mentioned cities. Apart from investors, public/tourists might also find this helpful.

THE DATA

In this section, the data and its source, description, preparation, and exploration are being discussed.

DATA SOURCE

Data source and quality is the heart of any data science problem as it has major effects on the method, prediction, and results of a problem. This project requires location data that describes hotels and restaurants around New York, Los Angeles, and Chicago. To extract this data, the "Foursquare" API was utilized as it provides valuable location data that powers some of the notable applications. A developer account was created in the "Foursquare" API which allows a person to extract data with certain limitations. Six major datasets were retrieved from the API i.e., Hotels and Restaurants in each city.

DATA DESCRIPTION

As discussed above, six datasets were retrieved in total. Two datasets per city for hotel and restaurants. For each dataset, we have the following attributes:

- Categories
- Number of Hotels & Restaurants within a radius
- Address
- Coordinates
- Distance from a center point
- Zip code
- City and State
- Cuisine type

The API also provided us with data such as ID, referral ID, venue page labels, delivery ID etc. which is of no significance in this project and hence they have been dropped from the DataFrame.

DATA PREPARATION

For each data set, a “Search” call was made to the Foursquare API with the search query “Hotel” & “Restaurants” by using the exact latitudinal and longitudinal coordinates. The radius of the search is set to 100000 and the data limit being 50 for each query (the max limit for a developer account).

DATA CLEANING

The aim was to get hotels in the downtown location. However, after the call was made results needed to be organized into a data frame.

Further cleaning was made to keep only data attributes related to the name and location of each venue as well as extracting the venues’ categories to clearly verify the results.

After results were organized and viewed in a Pandas data frame, it turned out that some results had categories other than “Hotel”. Therefore, these results were dropped from the data frame as our focus is on hotels only.

Second, the restaurants dataset was needed but this time the trending venues are the ones of interest to find the ideal cuisine , so an **“Explore”** call was made to the API to get a glimpse on which venues were most trending.

As in the case of restaurants, results were organized in a data frame and cleaned from any unnecessary features other than venues’ names and location related data. Then, venues’ categories were extracted and only restaurants were kept, and other venues were dropped from the data frame.

Since “Explore” call generates results that are trending now.

#Explore Call

```
url_LA2='https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET, VERSION, LA_lat, LA_lon, radius, limit_restaurants)
```

ATTRIBUTE SELECTION

For the intended analysis to be made, that is to find which areas have groups of hotels close to one another, and the number of trending restaurants in each area, two new data frames were created one for hotels and the other is for restaurants, each holding only the latitude and longitude coordinates of the original datasets as these are the features that some Machine Learning algorithms will focus on in our analysis as will be explained in the sections ahead.

METHODOLOGY

To clearly describe the methods of data analysis used in this project, a detailed explanation of our analysis will be presented here, showing the Machine Learning algorithms used, and how they contributed to the results as well as our statistical analysis of the opportunity of starting a restaurant in each of the cities discussed above.

LIBRARIES USED

The following libraries are used through the course of the project for modelling and analysis:

- NUMPY
- PANDAS
  - pd.set_option: Customize certain aspect of pandas
- JSON
  - JSON is text, written with JavaScript Object Notation
- JSON_NORMALIZE
  - Normalize semi structured JSON data into a flat table
- REQUESTS
  - Allows to send HTTP requests
- MATPLOTLIB.PLT
  - For plotting charts
- MATPLOTLIB.CM & MATPLOTLIB.COLORS
  - Built in color maps and color handling utilities
- SKLEARN.CLUSTER – KMEANS
  - Clustering of unlabeled data
- SKLEARN.NEIGHBORS – KNEIGHBORSCLASSIFIERS
- FOLIUM
  - To visualize a data on a map

DATA CLEANING CODES

As discussed in the earlier section, cleaning on data was performed to sort the required attributes from the extracted data. A few examples of data cleaning and its codes are given below.

#Filtering

```
filtered_columns2 = ['name', 'categories'] + [col for col in hotels_df2.columns if col.startswith('location.')] + ['id']
```

```
#Cleaning Columns

hotels_df_filtered2.columns = [column.split('.')[-1] for column in hotels_df_filtered2.columns]
```

Several other cleaning techniques have been used too and can be seen in the notebook.

FINDING THE LOCATION FOR ANALYSIS

The first step in finding the right spot for a restaurant near the city center of New York, Chicago and Los Angeles is knowing which areas have high density of tourists/population while being close to the city center or downtown.

The criteria mentioned above is already met when collecting the data from the Foursquare API by selecting a suitable radius for our search, the critical step here is to combine these hotels in groups (clusters) where each cluster will represent a hotspot where several hotels are located.

KMEANS CLUSTERING

To cluster the retrieved hotels into clusters, we need a Machine Learning clustering algorithm that can find similarities among data point and group them accordingly. Thus, K-Means Clustering algorithm was found to be most suitable due to its simplicity and effectiveness.

A Pandas data frame containing hotel's location coordinates was created, as mentioned earlier, and will serve as our input to the clustering algorithm, but the challenge is finding the appropriate number of clusters to group the data into. To overcome this issue, we visualized the hotels on a map centered around each city using Folium library and tried to inspect the data visually first.

Although other techniques are available to find the optimum number of clusters, such as, Elbow Method, but due to the relatively small number of data point visual inspection was enough where it was proposed that Three cluster would be suitable to group the data points so that number was used in the clustering algorithm.

```
#Clustering the hotels based on their locations:
kmeans2 = KMeans(n_clusters = 3, init ='k-means++')
kmeans2.fit(X2[X2.columns[1:3]]) # Compute k-means clustering.
X2['cluster_label'] = kmeans2.fit_predict(X2[X2.columns[1:3]])
```

The output from the clustering algorithm was added as a column to the locations data frame which was in turn merged back to the original hotels data frame, resulting in a complete table containing each hotel's name, location, cluster, etc.

However, the most important output was the centers on each cluster, which was also saved in another data frame containing the label of each cluster and its center's location coordinates (latitude and longitude).

```
#Coordinates of cluster centers.
centers2 = kmeans2.cluster_centers_
labels2 = kmeans2.predict(X2[X2.columns[1:3]]) # Labels of each point
centers2 = kmeans2.cluster_centers_
```

These two outputs were visualized on maps having the hotels in each cluster given a different color on the first map and the centers marked in larger circles.

LOCATING THE BEST SPOT

Once the hotel clusters have been located, the next step is to analyze competition in each area by finding the number of trending restaurants there. However, the first step in achieving this, is to classify the trending restaurants retrieved from Foursquare API into the clusters obtained from the K-Means algorithm.

K NEAREST NEIGHBORS CLASSIFICATION (KNN)

For the completion of this task, KNN classification algorithm was used to assign a class label to each restaurant in a data frame created earlier that contains only the coordinates data, but first a visualization of the restaurants locations is generated to try to predict the KNN results, and it was observed by comparison with the hotels locations, that at least one hotels' cluster did not have any restaurants.

KNN is a Supervised Machine Learning classification algorithm that assigns a label (class) to each data point according to the most frequent class among the nearest, user defined "K" number of neighbors. In other words, since the algorithm is supervised, we need to train the KNN model on a training dataset, and we need to specify the number of neighbors "K" that the classifier will use.

Therefore, we will use the clusters' centers' location data as a training set to tell the KNN model what location each class is center around, and we will use a "K" value of ONE as we want to assign the label of the closest single center to the data point in the restaurant's dataset.

```
#Preparing the data that will be used in KNN classification
Xhat_LA = restaurants_df_cleaned_LA.loc[:,['lat','lng']]


#Selecting classification features and target variable

X_train_LA = centers_df2[['c_lat', 'c_lon']].values
Y_train_LA = centers_df2['cluster_label']

#Classification based on the nearest center
k = 1

#Train Model and Predict
neigh_LA = KNeighborsClassifier(n_neighbors = k).fit(X_train_LA,Y_train_LA)

#Classify the restaurants based on the model created
yhat_LA = neigh_LA.predict(Xhat_LA)
```

The resulting class labels were added as a column in the original restaurants data frame and visualized on the map with assigning a different color to each class of restaurants similar to the hotels' output visualization (K-Means output).

EVALUATION

After grouping hotels in clusters, finding the center coordinates of each cluster, and assigning a class to each restaurant according to the nearest cluster center; it is now that an evaluation of opportunities is possible.

A Simple evaluation is to count the number of restaurants in each cluster to choose the one with the lowest level of competition, and then count the number of hotels in that cluster and compare it to the other clusters to evaluate opportunities. Simple value counting techniques were used in this evaluation.

FINDING THE IDEAL CUISINE

Once the location has been chosen, the question is "What should the new restaurant offer?"

To answer, we used a simple frequency counting technique against the restaurants data frame and inspected the top five venues as these venues: first, have high foot-traffic because they were obtained through an "Explore" call to Foursquare API, second, are most occurring among the retrieved restaurants.

NOTE: EVERY OF THE METHOD EXPLAINED ABOVE HAS BEEN APPLIED TO EACH CITY SEPARATELY TO ANALYZE

RESULTS & INFERENCES

In this segment, all results of the above explained methodology will be discussed and inferences will be derived.

CLUSTERING

After grouping the hotels into three clusters by the K-Means Clustering algorithm, the resulting data is visualized by assigning different color to each cluster.

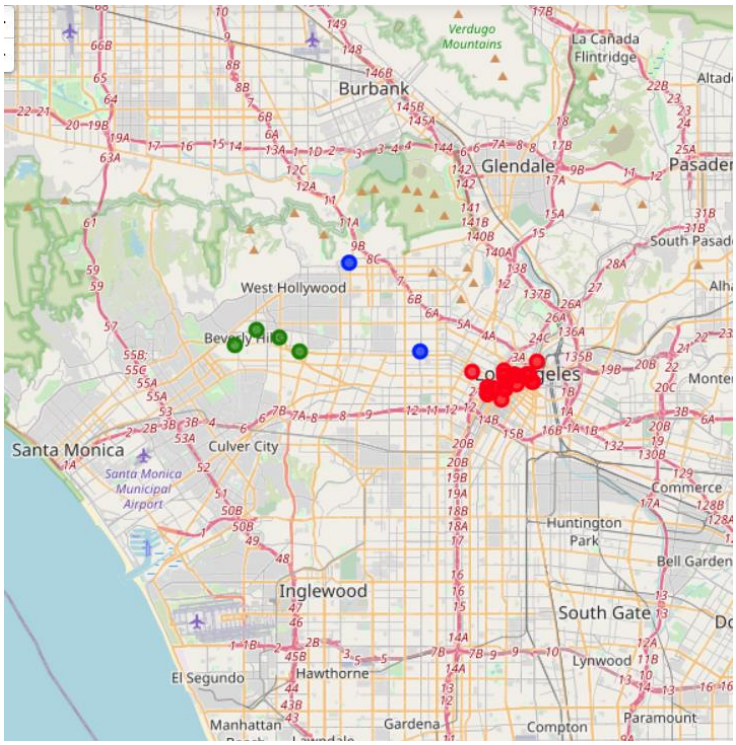

Fig 1: Hotel Cluster Map of Los Angeles

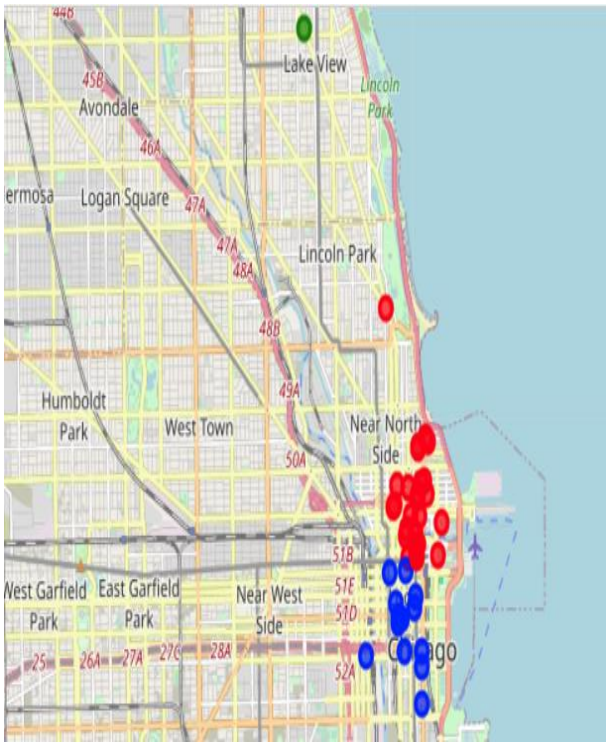

Fig *2: Hotel Cluster Map of Chicago*

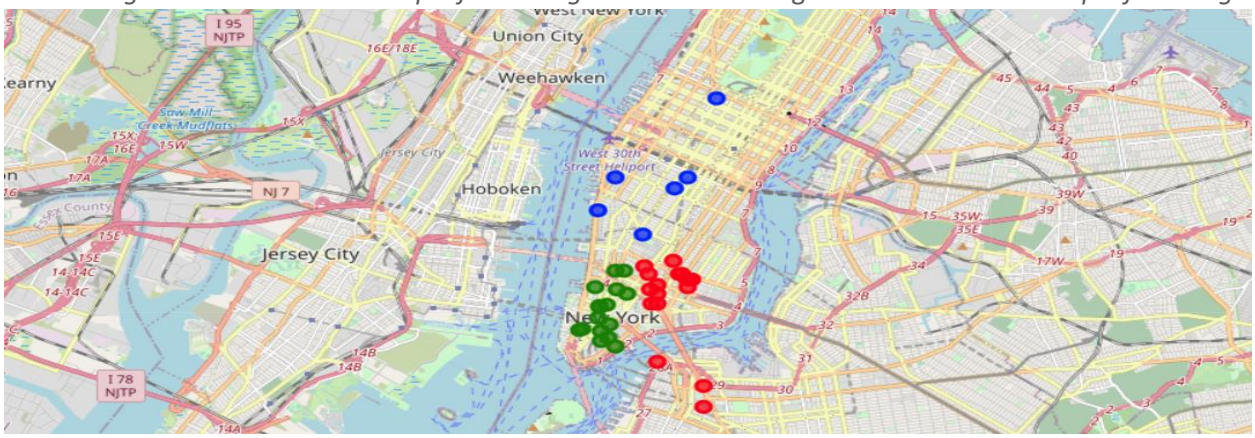

Fig 3: Hotel Cluster Map of New York

CLUSTER CENTERS

K-Means clustering algorithm also provides us with the centers of each clusters and its coordinates for analysis. These center coordinates are presented in the following table.

CITY	CLUSTERS	**LATITUDE**	**LONGITUDE**
NEW YORK	0/Red	40.6967	-73.9880
	1/Green	40.7246	-73.9946
	2/Blue	40.7127	-74.0083
LOS ANGELES	0/Red	34.0682	-118.3832
	1/Green	34.0501	-118.2529
	2/Blue	34.0633	-118.3006
CHICAGO	0/Red	41.8929	-87.6264
	1/Green	41.9478	-87.6572
	2/Blue	41.8781	-87.6294

Table 1: Cluster centers of each city

To visualize the above tabulated centers, a map with markers representing each center has been created using Folium library

.

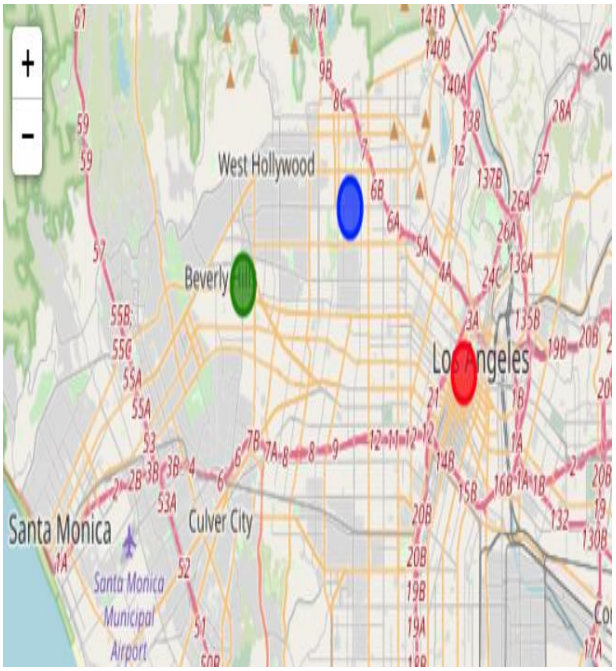

Fig 4: Center of Clusters Map (Los Angeles)

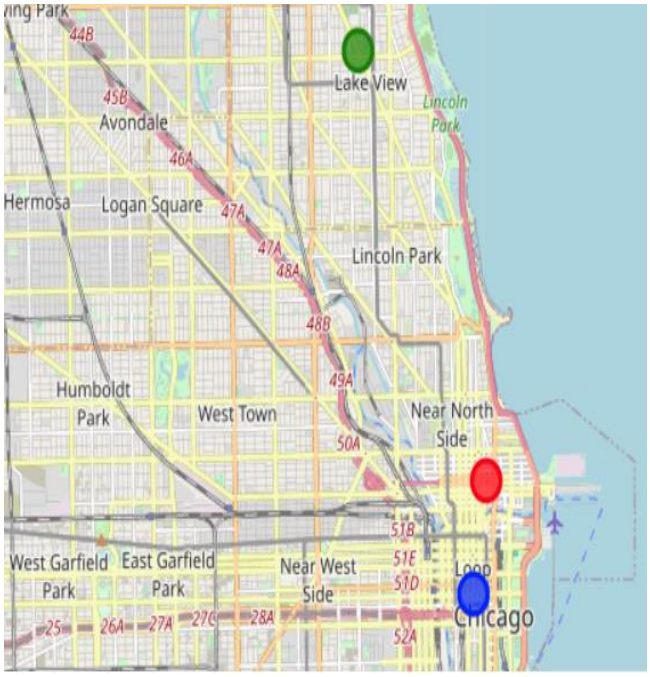

Fig 5: Center of Clusters Map (Chicago)

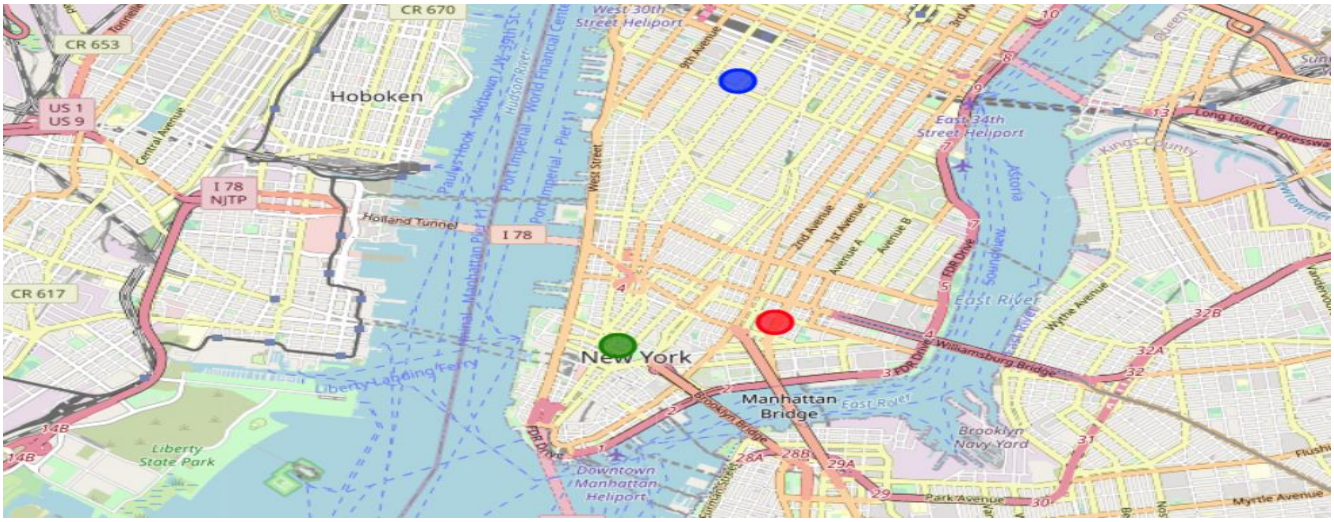

Fig 6: Center of Clusters Map (New York)

KNN CLASSIFICATION ALGORITHM

The folium output of KNN classification algorithm is shown below. The output shows each restaurant in a different color based on the class to which it is assigned.

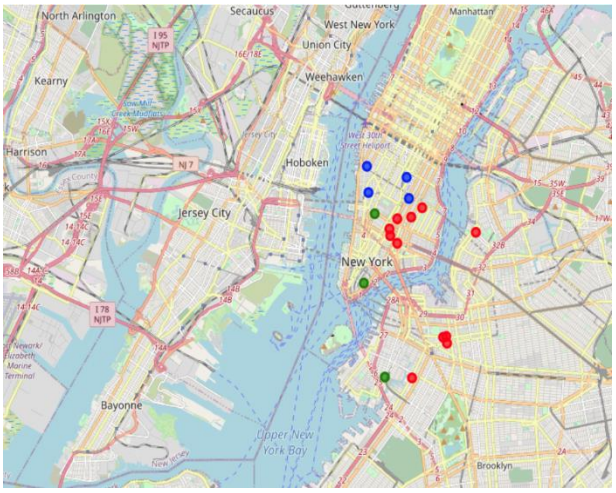

Fig 7: Restaurant Clusters Map of New York

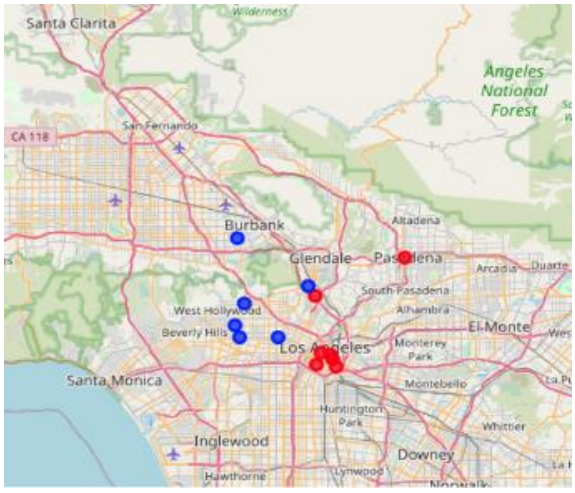

Fig 8: *Restaurant Clusters Map of Los Angeles*

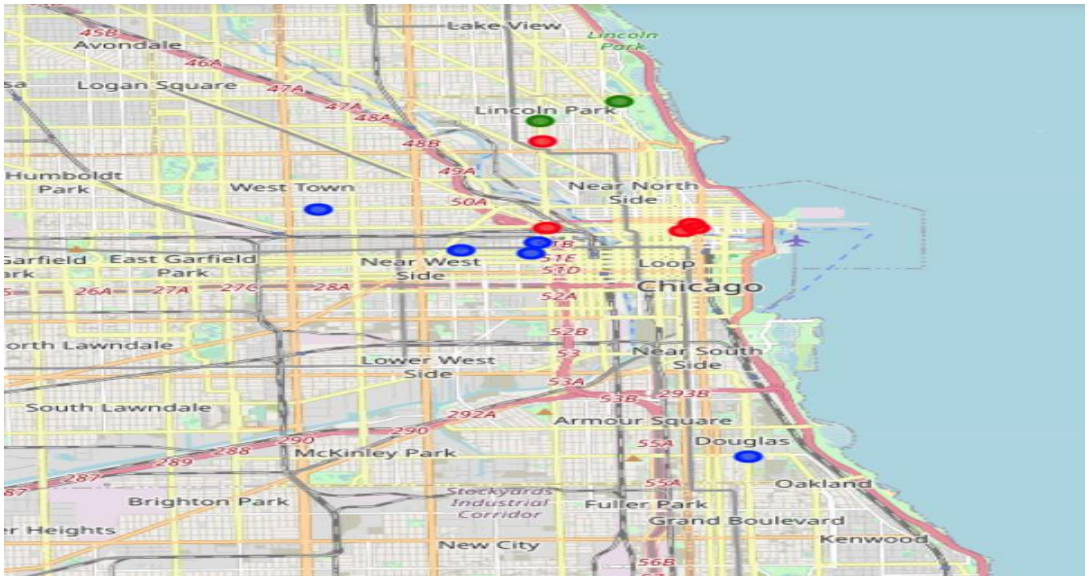

Fig 9: Restaurant Clusters Map of Chicago

THE COUNTS

RESTAURANTS:

For predicting completion levels, we counted the number of restaurants available in each map and the numbers are shown in the next table.

CITY	CLUSTER	NO. OF RESTAURANTS
NEW YORK	0/Red	5
	1/Green	13
	2/Blue	1
LOS ANGELES	0/Red	3
	1/Green	7
	2/Blue	3
CHICAGO	0/Red	5
	1/Green	2
	2/Blue	4

Table 2: Restaurant count of each cluster in each city

This evaluation will tell which cluster of hotels has the lowest number of restaurants and thus, a minimum level of completion.

HOTELS:

Since the area with the minimum number of restaurants was found, it is now convenient to verify if this area has enough hotels to consider it a worthy opportunity. Therefore, we counted the number of hotels in each area for every city.

CITY	CLUSTER	NO. OF HOTELS
NEW YORK	0/Red	3
	1/Green	19
	2/Blue	14
LOS ANGELES	0/Red	4
	1/Green	20
	2/Blue	1
CHICAGO	0/Red	27
	1/Green	1
	2/Blue	13

Table 3: Hotel count of each cluster in each city

IDEAL CUISINE:

Restaurants' categories were inspected, and we counted the frequency of each category to get an idea of the most occurring categories and thus, the restaurants receiving the highest demand, since the data frame used for this analysis already represents venues with the highest level of foot-traffic as explained before.

After counting the categories' frequency, we chose the top five restaurants' categories to visualize and analyze as this will guide our selection of the type of food our restaurant will supply.

The bar chart below represents top categories plotted against the number of restaurants of that category.

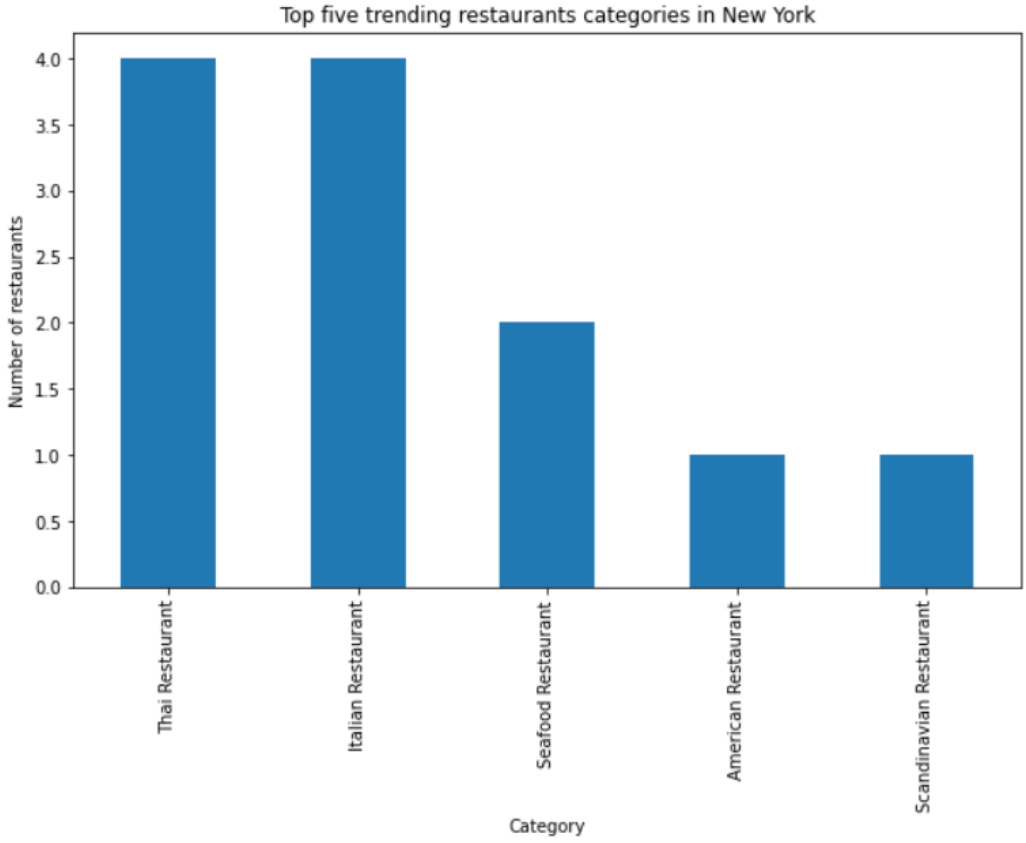

Fig 10: Bar chart for trending restaurant category in New York

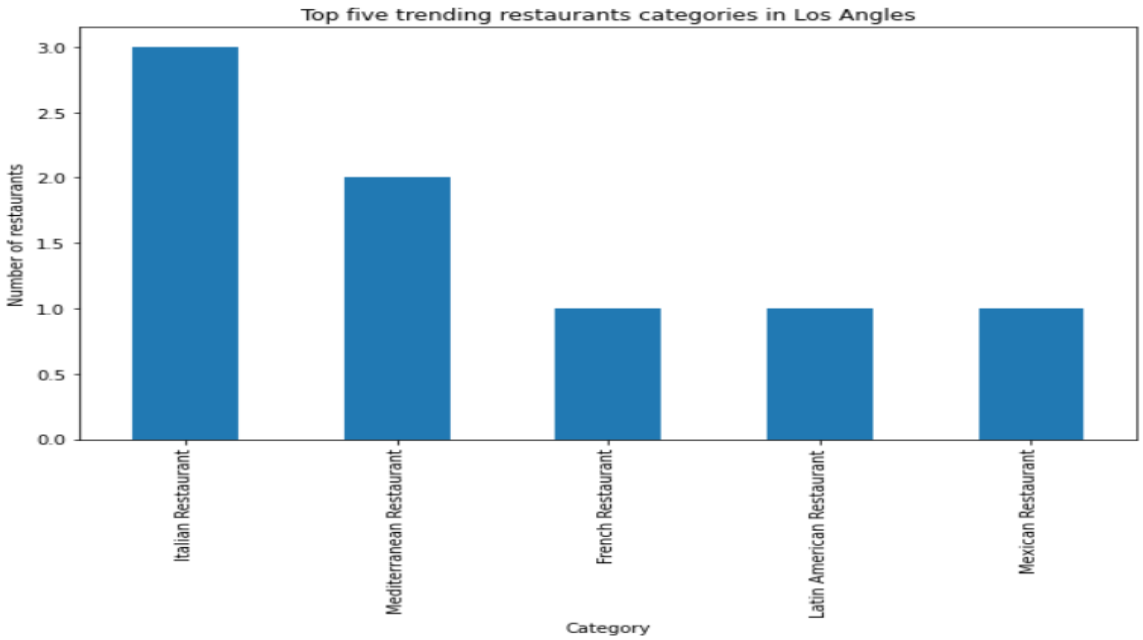

Fig 11: Bar chart for trending restaurant category in Los Angeles

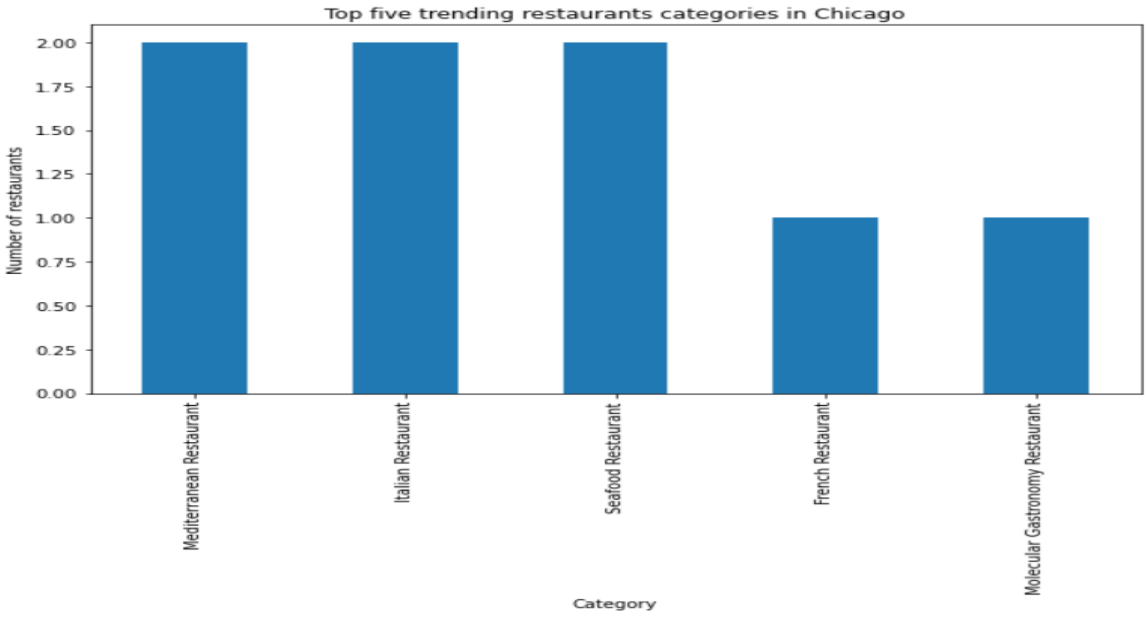

Fig 12: Bar chart for trending restaurant category in Chicago

The outputs and findings from this section is the sole of our analysis and decision making for the proposed business and will be further discussed in the next section.

DISCUSSION

The analysis performed in this project was intended to tell us where to open the proposed restaurant and what type of food it should supply.

After classifying the trending restaurants into clusters generated by K-Means algorithm, and counting the numbers in each cluster and city, we derive the following predictions:

NEW YORK:

We found that cluster 2 (shown in blue) has 14 hotels with just one restaurant, which can also be visualized by comparing the cluster maps of hotels and restaurants in New York city. This tells us that this area has the lowest level of competition.

In our analysis, we generated and visualized the center of each cluster as the point closest to all hotels in that cluster together. So, in the case of our proposed restaurant, the center of Cluster 2 located at (40.7127, -74.0083) would be the optimum spot, or as close to it as possible.

Finally, we tried to learn what food category was in demand in New York by looking at the top five frequent categories because of our explore call in the Foursquare API. Thai and Italian restaurant seems to be trending in the city. Also, the cluster 2 has no Italian restaurant so we can infer that opening an Italian restaurant in cluster 2 is ideal

LOS ANGELES:

Hotel cluster 1 (shown in green) has 20 hotels with seven restaurants (also visible in the folium maps). The other 2 clusters have very low number of hotels which means that the crowd for the business is too low to make comparison. Thus, we deduce that the cluster 1 is ideal. Also, the explore call from Foursquare shows that the trending category in Los Angeles is Italian. An internal comparison with the category and restaurant count shows that cluster 1 has just one Italian restaurant.

Therefore, opening an Italian restaurant in cluster 1 & closer to (34.0501 & -118.2529) coordinates seem optimal.

CHICAGO:

There are 27 hotels in cluster 0 (shown in red) and it has 5 restaurants in its radius. Alike Los Angeles, Chicago too has low number of hotels in the other two clusters thus making the target audience count low to make predictions. Therefore, we gather that cluster 0 is fitting. The Foursquare explore call shows that Mediterranean and Italian category of restaurants seems to be the most in demand in Chicago. Cluster 0 has one each of the above-mentioned categories.

So, it is safer to say that opening a restaurant closer to (41.8929 & -87.6264) coordinates with either of the categories would be ideal.

Altogether, when we consider all the three cities, we could make straight forward prediction about New York (the optimal location and ideal cuisine). Thus, the most ideal and safest prediction is about opening an Italian restaurant in the city of New York closer to the center of the cluster 2 or closer to (40.7127, -74.0083) coordinates.

CONCLUSION

In this project our aim was to find the best location for a restaurant in three of the most populated cities in the United States targeting tourists, and the food type it should provide. To find that we retrieved data from Foursquare API about hotels and restaurants in the desired location.
Hotels' data was analyzed using K-Means Clustering algorithm to group the hotels in three clusters and find the center of each cluster. This was done to find the areas with a high density of hotels and thus considered as touristic areas.
Restaurants' data on the other hand was used as a target dataset for KNN Classification algorithm to see in which clusters the trending restaurants are located. This helped us find that a particular cluster in each city had the lowest level of competition as it had no to only a few restaurants.
Finally, we looked at the food type in demand the most by counting the frequency each restaurant category in our data occurred, and we found that Italian food was in demand and restaurants with Italian food menus had the highest levels of foot-traffic.

APPENDIX:

Python codes of the project:

PREDICTING OPTIMAL LOCATION TO OPEN RESTAURANTS TARGETING TOURISTS IN TOP THREE POPULATED CITIES IN UNITED STATES OF AMERICA.

In [ ]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [ ]:

```
import numpy as np

import pandas as pd
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

import json

import requests
from pandas.io.json import json_normalize

import matplotlib.cm as cm
import matplotlib.colors as colors
import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

try:
    import folium
except:
    !import folium
```

Getting the hotels data around New York city center and performing needed analysis

In this part, we will call the Foursquare API and search for hotels around New York city center. Then we will cluster the resulting venues using K-Means clustering to find areas were groups of hotels are located. Finally, we will find the centers of these areas and organize them in a data frame for later analysis.

In [ ]:

```
#setting the needed information to call Foursquare API

CLIENT_ID = 'LQA5A1YTM25KN0TQT2GILWCCP2JLPMPQN20YN4ZY0QOTT2ZH' # your
Foursquare ID
```

```
CLIENT_SECRET = '0IXCMPXJLX3LFJZUOH33ZEQHRG2JIZQF3R4KKMPIJDWQNHMV' # your
Foursquare Secret
VERSION = '20210415' # Foursquare API version
```

In [ ]:

```
new_york_lat = 40.7128
new_york_lon = -74.0060
search_query = 'Hotel'
radius = 100000
limit_hotels = 50
```

In [ ]:

```
url =
'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll
={},{}&v={}&query={}&radius={}&limit={}'.format(CLIENT_ID, CLIENT_SECRET,
new_york_lat, new_york_lon, VERSION, search_query, radius, limit_hotels)
url
```

In [ ]:

```
results = requests.get(url).json()
```

In [ ]:

```
hotels = results['response']['venues']

# tranform venues into a dataframe
hotels_df = pd.json_normalize(hotels)
hotels_df.head()
```

In [ ]:

```
filtered_columns = ['name', 'categories'] + [col for col in hotels_df.columns
if col.startswith('location.')] + ['id']
hotels_df_filtered = hotels_df.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# filter the category for each row
hotels_df_filtered['categories'] =
hotels_df_filtered.apply(get_category_type, axis=1)

# clean column names by keeping only last term
hotels_df_filtered.columns = [column.split('.')[-1] for column in
hotels_df_filtered.columns]
```

```
hotels_df_filtered.shape
```

In [ ]:

```
# keep only columns that include venue name, and anything that is associated
with location
filtered_columns = ['name', 'categories'] + [col for col in hotels_df.columns
if col.startswith('location.')] + ['id']
hotels_df_filtered = hotels_df.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# filter the category for each row
hotels_df_filtered['categories'] =
hotels_df_filtered.apply(get_category_type, axis=1)

# clean column names by keeping only last term
hotels_df_filtered.columns = [column.split('.')[-1] for column in
hotels_df_filtered.columns]

hotels_df_filtered
```

In [ ]:

```
hotels_df_filtered = hotels_df_filtered[hotels_df_filtered.categories ==
'Hotel']
hotels_df_filtered.head(10)
```

In [ ]:

```
hotels_df_filtered.reset_index(drop=True, inplace=True)
hotels_df_filtered.head(10)
```

In [ ]:

```
hotels_map = folium.Map(location=[new_york_lat, new_york_lon], zoom_start=15)
# generate map centred around London

# add the hotels as blue circle markers
for lat, lng, label in zip(hotels_df_filtered.lat, hotels_df_filtered.lng,
hotels_df_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        color='blue',
        popup=label,
        fill = True,
```

```
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(hotels_map)

# display map
hotels_map
```

In [ ]:

```
hotels_df_filtered.shape
```

In [ ]:

```
#prepare hotels dataframe for clustering

#adding an ID number to each venue will help in re-assembling the data after
clustering
id_n = list(range(1,len(hotels_df_filtered)+1))
hotels_df_filtered['id_n'] = id_n
hotels_df_filtered.head()
```

In [ ]:

```
X=hotels_df_filtered.loc[:,['id_n','lat','lng']]
X.head(10)
```

In [ ]:

```
#clustering the hotels based on their locations
kmeans = KMeans(n_clusters = 3, init ='k-means++') # hotels will be clustered
in 3 groups
kmeans.fit(X[X.columns[1:3]]) # Compute k-means clustering.
X['cluster_label'] = kmeans.fit_predict(X[X.columns[1:3]])
centers = kmeans.cluster_centers_ # Coordinates of cluster centers.
labels = kmeans.predict(X[X.columns[1:3]]) # Labels of each point
print(centers) #to show coordinates of the center for each cluster
X.head(10)
```

In [ ]:

```
X = X[['id_n','cluster_label']]
clustered_hotels = hotels_df_filtered.merge(X, left_on='id_n',
right_on='id_n')
clustered_hotels.head()
```

In [ ]:

```
#count number of hotels in each area
hotel_count =
clustered_hotels['cluster_label'].value_counts().rename_axis('cluster').reset
_index(name='counts')
hotel_count
```

In [ ]:

```
#map showing each hotel with different color based on the cluster
map_clusters = folium.Map(location=[new_york_lat, new_york_lon],
zoom_start=15)

# add markers to the map
```

```
marker_colors = ['red','green','blue']
for lat, lon, poi, cluster in zip(clustered_hotels['lat'],
clustered_hotels['lng'], clustered_hotels['name'],
clustered_hotels['cluster_label']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster),
parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=marker_colors[cluster],
        fill=True,
        fill_color=marker_colors[cluster],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters #show map
```

In [ ]:

```
centers_df = pd.DataFrame(centers, columns =['c_lat', 'c_lon'])
cluster_label = list(range(0,3))
centers_df['cluster_label'] = cluster_label
centers_df
```

In [ ]:

```
#map of centers
map_centers = folium.Map(location=[new_york_lat, new_york_lon],
zoom_start=15)

# add markers to the map in different colors according to cluster
marker_colors = ['red','green','blue']
for lat, lon, poi, cluster in zip(centers_df['c_lat'], centers_df['c_lon'],
centers_df['cluster_label'], centers_df['cluster_label']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster),
parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=10,
        popup=label,
        color=marker_colors[cluster],
        fill=True,
        fill_color=marker_colors[cluster],
        fill_opacity=0.7).add_to(map_centers)

map_centers #display the map showing 3 markers, one for each cluster
```

Getting the restaurants data from Foursquare around New York city center and analysing it.

In this section, we will call foursquare API again and call for trending venues around New York city center using the "explore" call, then we will clean the data, keep restaurants only, and classify these restaurants to one of the clusters created earlier using KNN classification.

To do this we will use the centers data created before as a training set for KNN, then we will predict the cluster to which each restaurant belongs to by finding the nearest center.

In [ ]:

```
#creating the url to call on foursquare API

limit_restaurants = 100

url2 =
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&
v={}&ll={},{}&radius={}&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    new_york_lat,
    new_york_lon,
    radius,
    limit_restaurants)
url2
```

In [ ]:

```
results2 = requests.get(url2).json()
```

In [ ]:

```
#getting the needed data and put it in a pandas dataframe
restaurants = results2['response']['groups'][0]['items']

restaurants_df = json_normalize(restaurants) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.location.lat',
'venue.location.lng']
restaurants_df = restaurants_df.loc[:, filtered_columns]

# filter the category for each row
restaurants_df['venue.categories'] = restaurants_df.apply(get_category_type,
axis=1)

# clean columns
restaurants_df.columns = [col.split(".")[-1] for col in
restaurants_df.columns]

restaurants_df
```

In [ ]:

```
#keeping only the venues with categories containing "Restaurant"
restaurants_df_cleaned =
restaurants_df[restaurants_df['categories'].str.contains('Restaurant')].reset
_index(drop=True)
print(restaurants_df_cleaned.shape) #number of restaurants
restaurants_df_cleaned.head()
```

In [ ]:

```
#creating a map of restaurants
restaurants_map = folium.Map(location=[new_york_lat, new_york_lon],
zoom_start=15) # generate map centred around London

# add the hotels as yellow circle markers
for lat, lng, label in zip(restaurants_df_cleaned.lat,
restaurants_df_cleaned.lng, restaurants_df_cleaned.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        color='brown',
        popup=label,
        fill = True,
        fill_color='borwn',
        fill_opacity=0.6
    ).add_to(restaurants_map)

# display map
restaurants_map
```

In [ ]:

```
#preparing the data that will be used in KNN classification
Xhat=restaurants_df_cleaned.loc[:,['lat','lng']] #classification based on the
location coordinates
print(Xhat.shape) #number of restaurants
Xhat.head(10) #show dataframe
```

In [ ]:

```
#the centers dataframe to be used for training KNN
centers_df
```

In [ ]:

```
#selecting classification features and target variable
X_train = centers_df[['c_lat', 'c_lon']].values
Y_train = centers_df['cluster_label']
```

In [ ]:

```
#importing KNN library
from sklearn.neighbors import KNeighborsClassifier
```

In [ ]:

```
k = 1 #classification based on the nearest center
#Train Model and Predict
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,Y_train)
neigh
```

In [ ]:

```
#classify the restaurants based on the model created
yhat = neigh.predict(Xhat)
yhat[0:5]
```

In [ ]:

```
#putting the results in a dataframe
```

```
KNN_results = pd.DataFrame(yhat, columns=['class'])
KNN_results
```

In [ ]:

```
#joining the results with the restaurants dataframe
restaurants_classified = pd.concat([restaurants_df_cleaned, KNN_results],
axis=1, sort=False)
```

In [ ]:

```
#view resulting dataframe
restaurants_classified
```

In [ ]:

```
#show restaurante in each area in different color
map_rest_class = folium.Map(location=[new_york_lat, new_york_lon],
zoom_start=15)

# add markers to the map
marker_colors = ['red','green','blue']
for lat, lon, poi, cluster in zip(restaurants_classified['lat'],
restaurants_classified['lng'], restaurants_classified['name'],
restaurants_classified['class']):
    label = folium.Popup(str(poi) + ' Class ' + str(cluster),
parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=marker_colors[cluster],
        fill=True,
        fill_color=marker_colors[cluster],
        fill_opacity=0.7).add_to(map_rest_class)

map_rest_class
```

Finding the suitable spot to open restaurant around New York.

In this section we will see what category of restaurants is most trending and desired in center of New York. Since our restaurants data represent venues with highest foot-traffic then we will look at the frequency of restaurant category repetition as this are the venues recieving customers.

Then we will look at the number of restaurants in each of the three areas to choose the cluster of hotels with the minimum level of competion, as well as the number of hotels in that cluster to correctly evaluate the opportunity.

In [ ]:

```
#count number of restaurants in each area
restaurants_count =
restaurants_classified['class'].value_counts().rename_axis('class').reset_ind
ex(name='counts')
restaurants_count
```

In [ ]:

```
#counting number of trending restaurants of each unique category
trending_restaurants =
restaurants_classified['categories'].value_counts().rename_axis('categories')
.reset_index(name='counts')
trending_restaurants
```

In [ ]:

```
#keeping the five most frequent categories
trending_restaurants = trending_restaurants.head()
```

In [ ]:

```
#show top five categories
trending_restaurants
```

In [ ]:

```
#make "categories" as index for visiualization
trending_restaurants = trending_restaurants.set_index('categories')
trending_restaurants
```

In [ ]:

```
#importing bar charts library
import matplotlib as mpl
import matplotlib.pyplot as plt
```

In [ ]:

```
#plotting the top five restaurant categories
trending_restaurants.plot(kind='bar', figsize=(10, 6), legend=None)

plt.xlabel('Category') # add to x-label to the plot
plt.ylabel('Number of restaurants') # add y-label to the plot
plt.title('Top five trending restaurants categories in New York') # add title
to the plot

plt.show()
```

The above set of codes is repeated with the other 2 cities (Chicago and Los Angeles). Since the codes are almost repeating and is extensive the codes are not pasted here instead the jupyter notebook file is attached.

REFERENCES:

1) O'Reilly Python for Data Analysis
2) O'Reilly Python Data Science Handbook
3) O'Reilly Introduction to Machine Learning with Python
4) https://scikit-learn.org/stable/
5) https://pythonprogramminglanguage.com/how-is-the-k-nearest-neighbor-algorithm-different-from-k-means-clustering/
6) https://developer.foursquare.com/
7) https://towardsdatascience.com/