EXPLORATORY ANALYSIS OF GEOLOCATIONAL DATA

PROJECT BY: Vasudharini.V.J(22011101120), Rithanya.M(22011101090)

OVERVIEW:

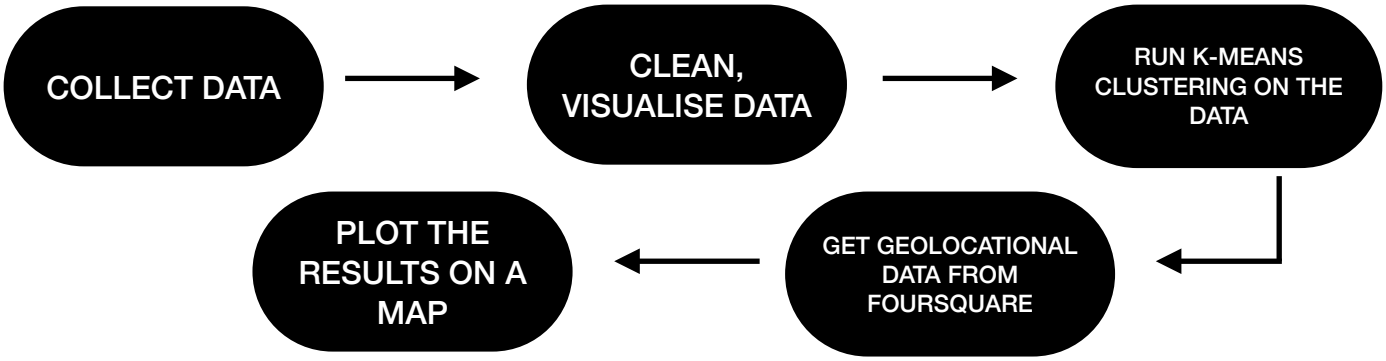
This project involves the use of K-Means Clustering and affinity propogation to find the best accommodation for students in Hyderabad (or any other city of your choice) by classifying accommodation for incoming students on the basis of their preferences on amenities, budget and proximity to the location.

METHODOLOGY:

This paper proposed a system to understand Geolocational data for students seeking accommodation. The geolocational data fetched from Foursquare API (Application programming interface) helps to analyse the students proximity to amenities. This information provides a thorough study of student preferences and tastes.

Steps involved are:

- Fetch Datasets from the relevant locations (Data Collection)
- Clean the Datasets to prepare them for analysis. (Data Cleaning via Pandas)
- Visualise the data using boxplots. (Using Matplotlib /Seaborn /Pandas)
- Fetch Geolocational Data from the Foursquare API. (REST APIs)
- Use K-Means Clustering to cluster the locations (Using ScikitLearn)
- Verify the same with Affinity propogation (Using ScikitLearn)
- Present findings on a map. (Using Folium/Seaborn)



REQUIREMENT TOOLS:

- Python
- For data - numpy and pandas package.
- For plotting - matplotlib package & seaborn packages
- For geospatial - geopy, folium.
- For machine learning - sklearn (preprocessing and cluster) scipy,
- For deep learning - minisom

CODE:

DATA COLLECION:

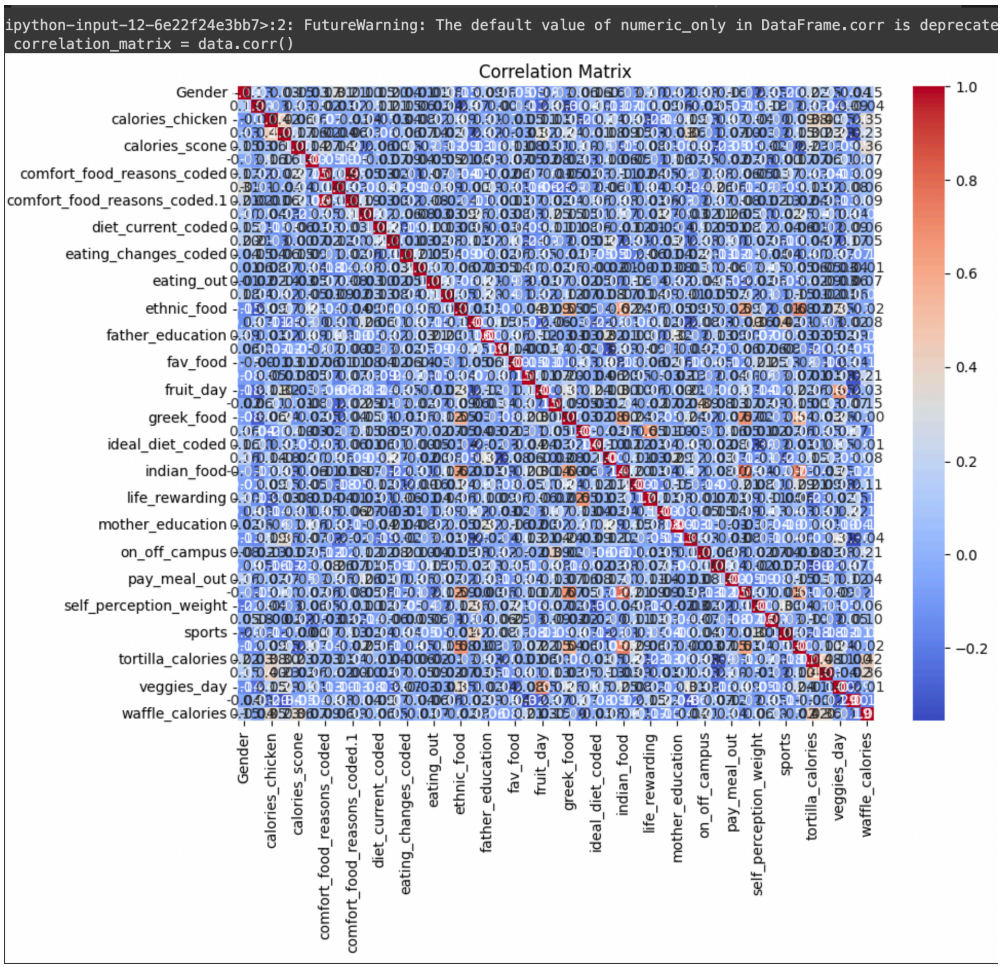
```
import pandas as pd
data=pd.read_csv("/content/drive/MyDrive/  Exploratory Analysis of Geolocational Data/food_coded.csv")
data
```

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	con
0	2.4	2	1	430	NaN	315.0	1	none	we dont have comfort	
1	3.654	1	1	610	3.0	420.0	2	chocolate, chips, ice cream	Stress, bored, anger	
2	3.3	1	1	720	4.0	420.0	2	frozen yogurt, pizza, fast food	stress, sadness	
3	3.2	1	1	430	3.0	420.0	2	Pizza, Mac and cheese, ice cream	Boredom	

DATA CLEANING:

TRYING FEATURE SELECTION USING CORRELATION MATRIX:

```
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True,
cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
threshold = 0.5
high_correlation_features =
correlation_matrix[abs(correlation_matrix) > threshold]
high_correlation_features =
high_correlation_features[high_correlation_features !=
1].dropna(axis=1, how='all')
selected_features =
high_correlation_features.columns.tolist()

print("Selected Features:")
print(selected_features)
```



Data Cleaning

```
[13] selected_features
```

```
['comfort_food_reasons_coded',
 'comfort_food_reasons_coded.1',
 'ethnic_food',
 'fruit_day',
 'greek_food',
 'healthy_feeling',
 'indian_food',
 'life_rewarding',
 'persian_food',
 'thai_food',
 'veggies_day']
```

The selected features are not the most suitable for the following analysis, as the data is smaller and a known domain going forward with manual selection of the features from the data:

Feature selection 2:

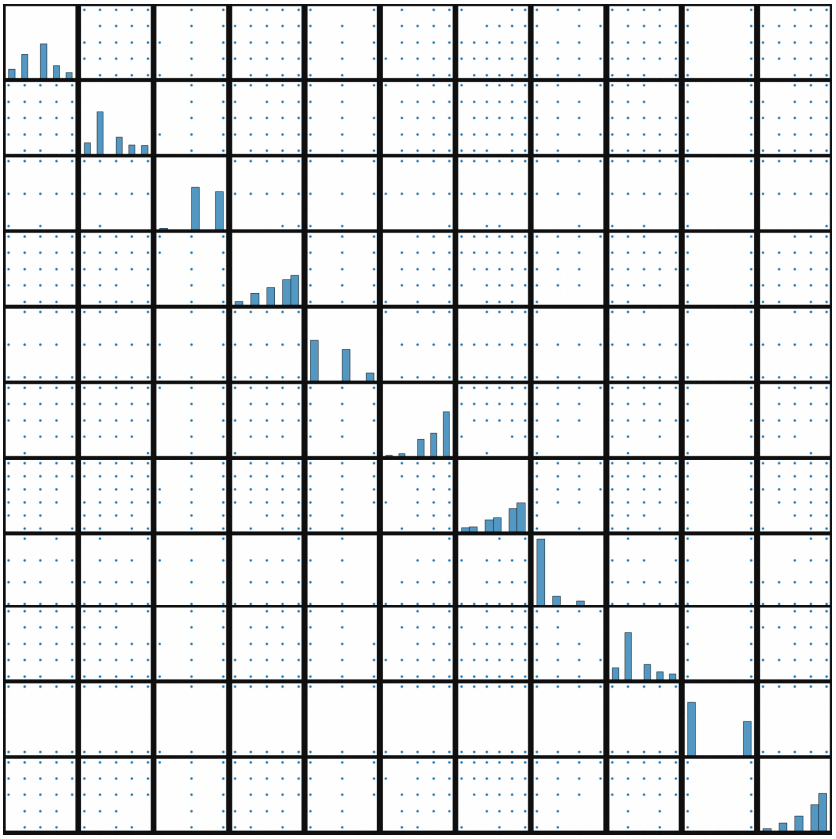
```
column=['cook','eating_out','employment','ethnic_food',
'exercise','fruit_day','income','on_off_campus','pay_meal_out
','sports','veggies_day']
```

	cook	eating_out	employment	ethnic_food	exercise	fruit_day	income	on_off_campus	pay_meal_out	sports	veggies_da
0	2.0	3	3.0	1	1.0	5	5.0	1.0	2	1.0	
1	3.0	2	2.0	4	1.0	4	4.0	1.0	4	1.0	
2	1.0	2	3.0	5	2.0	5	6.0	2.0	3	2.0	
3	2.0	2	3.0	5	3.0	4	6.0	1.0	2	2.0	
4	1.0	2	2.0	4	1.0	4	6.0	1.0	4	1.0	
...	...	...	...	...	...	...	...	...	...	...	
120	3.0	2	1.0	4	2.0	5	4.0	3.0	4	1.0	
121	3.0	4	3.0	3	2.0	4	2.0	1.0	4	NaN	
122	3.0	3	3.0	5	2.0	4	2.0	1.0	4	2.0	
123	3.0	5	2.0	2	1.0	5	4.0	1.0	3	2.0	
124	NaN	1	2.0	3	2.0	3	5.0	1.0	3	2.0	

```
d=data[column]
d
```

DATA EXPLORATION AND VISUALISATION:

```
import seaborn as sns
sns.pairplot(d)
```

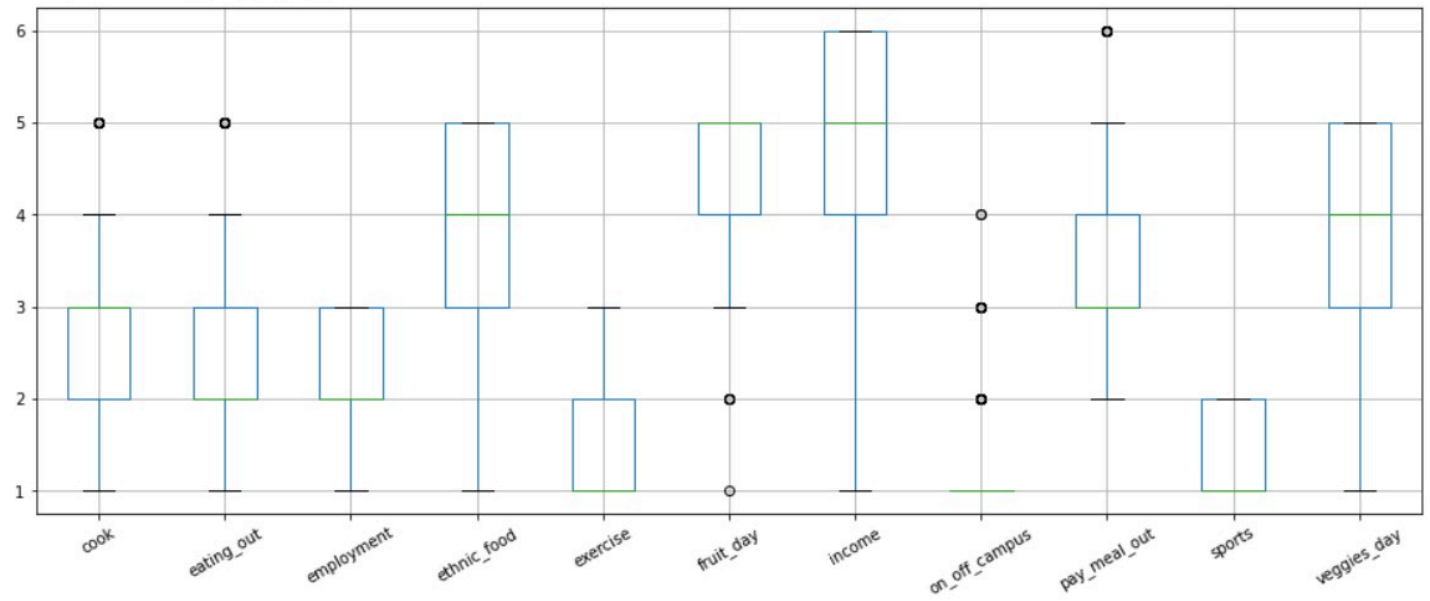


BOXPLOT OF DATASET:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline
ax=d.boxplot(figsize=(16,6))
ax.set_xticklabels(ax.get_xticklabels(),rotation=30)
d.shape
s=d.dropna()
```

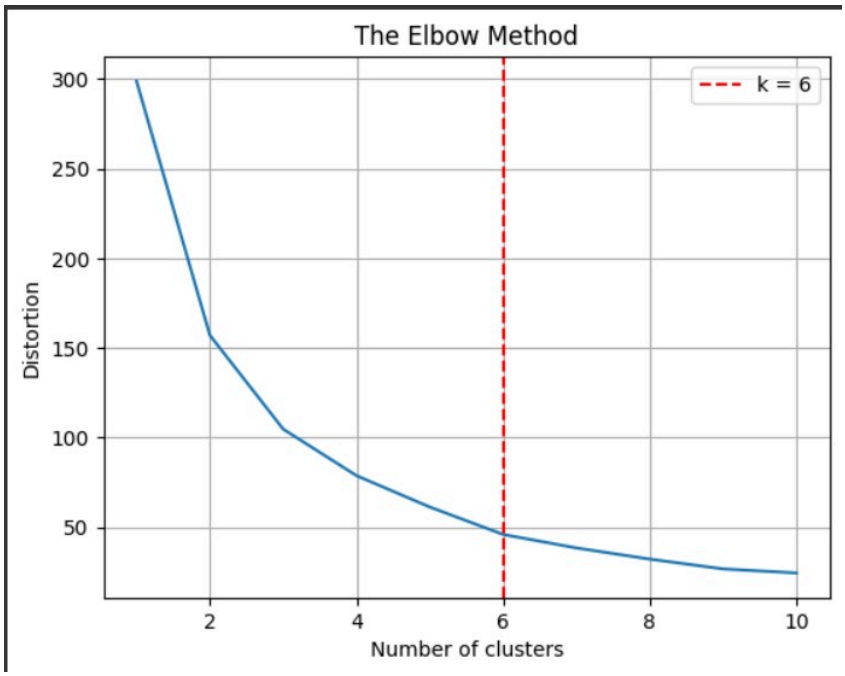
RUN K-MEANS CLUSTERING ON THE DATA:

```
## for data
import numpy as np
import pandas as pd
## for plotting
import matplotlib.pyplot as plt
import seaborn as sns
```




```
## for geospatial
import folium
import geopy
## for machine learning
from sklearn import preprocessing, cluster
import scipy
## for deep learning
import minisom
f=['cook','income']
X = s[f]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
        model = cluster.KMeans(n_clusters=i, init='k-means++',
max_iter=300, n_init=10, random_state=0)
        model.fit(X)
        distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100
for i
     in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
```

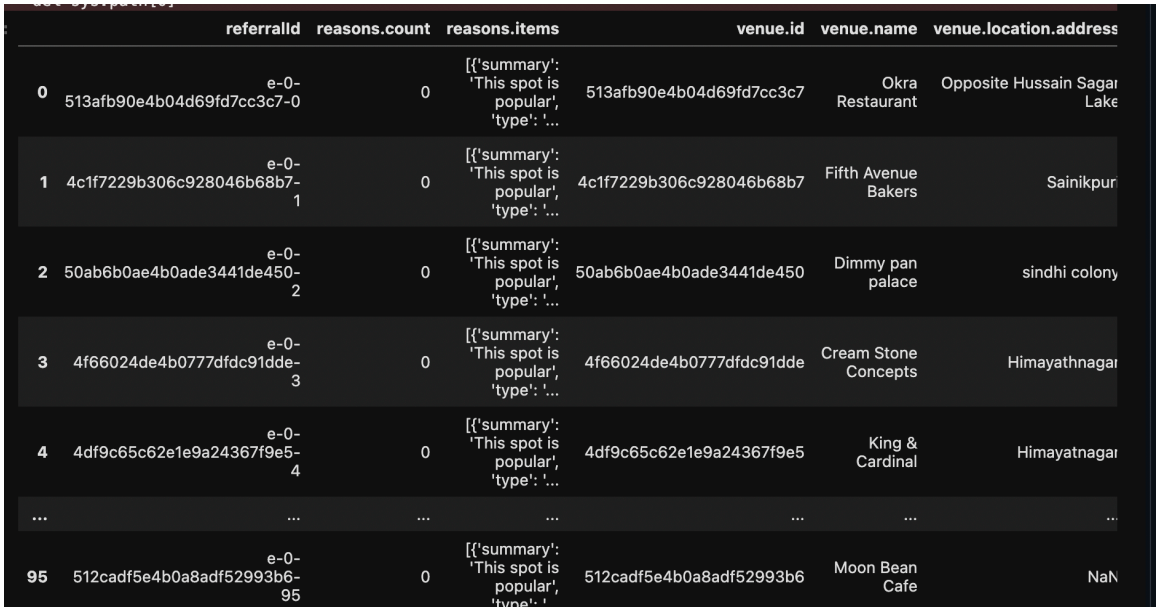
```
ax.legend()
ax.grid(True)
plt.show()
```



GET GEOLOCATIONAL DATA:

```
from pandas.io.json import json_normalize
import folium
from geopy.geocoders import Nominatim
import requests
CLIENT_ID =
"KTCJJ2YZ2143QHEZ2JAQS4FJIO5DLSDO0YN4YBXPMI5NKTEF" # your
Foursquare ID
CLIENT_SECRET =
"KNG2LO22BPLHN1E3OAHWLYQ5PQBN14XYZMEMAS0CPJEJKOTR" # your
Foursquare Secret
VERSION = '20200316'
LIMIT = 10000
url = 'https://api.foursquare.com/v2/venues/explore?
&client_id={}&client_secret={}&v={}&ll={},{}&radius={}
&limit={}'.format(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
```

```
    17.448372, 78.526957,
    30000,
    LIMIT)
results = requests.get(url).json()
results

venues = results['response']['groups'][0]['items']
nearby_venues = json_normalize(venues)

nearby_venues
```

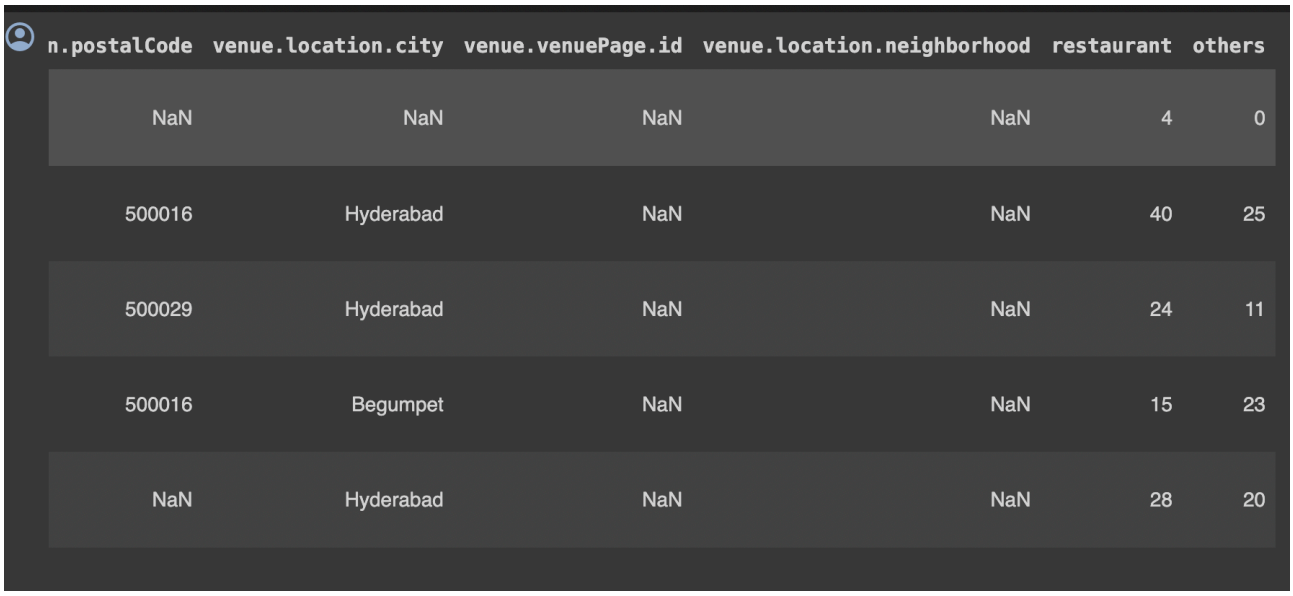


	referralId	reasons.count	reasons.items	venue.id	venue.name	venue.location.address
0	e-0-513afb90e4b04d69fd7cc3c7-0	0	[{'summary': 'This spot is popular', 'type': '...	513afb90e4b04d69fd7cc3c7	Okra Restaurant	Opposite Hussain Sagar Lake
1	e-0-4c1f7229b306c928046b68b7-1	0	[{'summary': 'This spot is popular', 'type': '...	4c1f7229b306c928046b68b7	Fifth Avenue Bakers	Sainikpuri
2	e-0-50ab6b0ae4b0ade3441de450-2	0	[{'summary': 'This spot is popular', 'type': '...	50ab6b0ae4b0ade3441de450	Dimmy pan palace	sindhi colony
3	e-0-4f66024de4b0777dfdc91dde-3	0	[{'summary': 'This spot is popular', 'type': '...	4f66024de4b0777dfdc91dde	Cream Stone Concepts	Himayathnagar
4	e-0-4df9c65c62e1e9a24367f9e5-4	0	[{'summary': 'This spot is popular', 'type': '...	4df9c65c62e1e9a24367f9e5	King & Cardinal	Himayatnagar
...	...	...	...	...	...	...
95	e-0-512cadf5e4b0a8adf52993b6-95	0	[{'summary': 'This spot is popular', 'type': '...	512cadf5e4b0a8adf52993b6	Moon Bean Cafe	NaN

ADDING TWO MORE COLUMNS RESTAURANT AND OTHERS

1. Restaurant: Number of Restaurant in the radius of 20 km
2. Others: Number of Gyms, Parks, etc in the radius of 20 km

```
resta=[]
oth=[]
for lat,long in
zip(nearby_venues['venue.location.lat'],nearby_venues['venue.
location.lng']):
    url = 'https://api.foursquare.com/v2/venues/explore?
&client_id={}&client_secret={}&v={}&ll={},{}&radius={}
&limit={}'.format(
      CLIENT_ID,
      CLIENT_SECRET,
      VERSION,
      lat,long,
      1000,
```



n.postalCode	venue.location.city	venue.venuePage.id	venue.location.neighborhood	restaurant	others
NaN	NaN	NaN	NaN	4	0
500016	Hyderabad	NaN	NaN	40	25
500029	Hyderabad	NaN	NaN	24	11
500016	Begumpet	NaN	NaN	15	23
NaN	Hyderabad	NaN	NaN	28	20

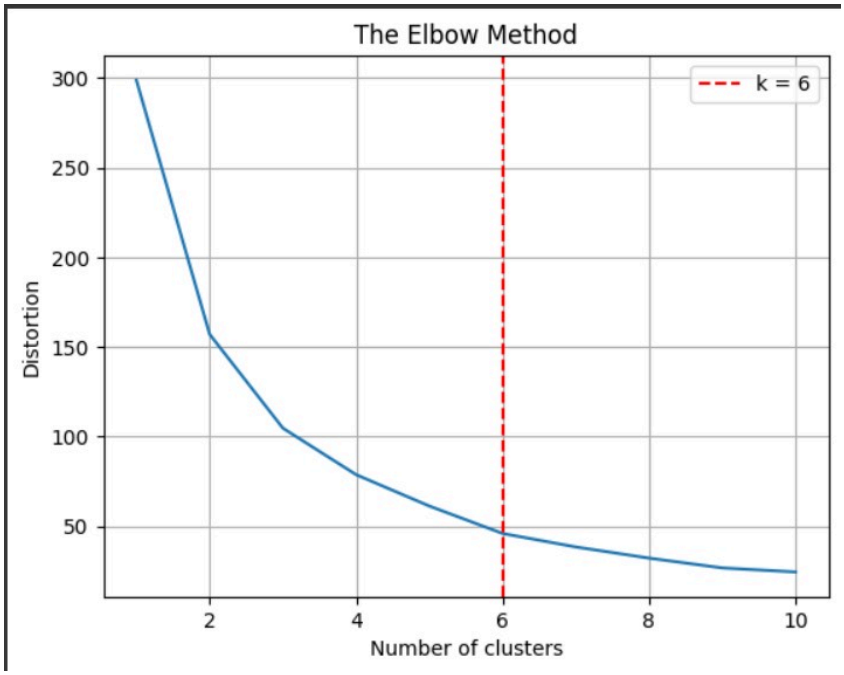
```
      100)
    res = requests.get(url).json()
    venue = res['response']['groups'][0]['items']
    nearby_venue = json_normalize(venue)
    df=nearby_venue['venue.categories']

    g=[]
    for i in range(0,df.size):
      g.append(df[i][0]['icon']['prefix'].find('food'))
    co=0
    for i in g:
      if i>1:
        co+=1
    resta.append(co)
    oth.append(len(g)-co)

nearby_venues['restaurant']=resta
nearby_venues['others']=oth
nearby_venues
```

CHANGING THE COLUMN NAME:

```
lat=nearby_venues['venue.location.lat']
long=nearby_venues['venue.location.lng']
```

RUN K-MEANS CLUSTERING ON THE DATASET, WITH THE OPTIMAL K VALUE USING ELBOW METHOD:

```
f=['venue.location.lat','venue.location.lng']
X = nearby_venues[f]
max_k = 10
## iterations
distortions = []
for i in range(1, max_k+1):
    if len(X) >= i:
       model = cluster.KMeans(n_clusters=i, init='k-means++',
max_iter=300, n_init=10, random_state=0)
       model.fit(X)
       distortions.append(model.inertia_)
## best k: the lowest derivative
k = [i*100 for i in np.diff(distortions,2)].index(min([i*100
for i
     in np.diff(distortions,2)]))
## plot
fig, ax = plt.subplots()
ax.plot(range(1, len(distortions)+1), distortions)
ax.axvline(k, ls='--', color="red", label="k = "+str(k))
ax.set(title='The Elbow Method', xlabel='Number of clusters',
       ylabel="Distortion")
ax.legend()
ax.grid(True)
plt.show()
```



```
city = "Hyderabad"
## get location
locator = geopy.geocoders.Nominatim(user_agent="MyCoder")
location = locator.geocode(city)
print(location)
## keep latitude and longitude only
location = [location.latitude, location.longitude]
print("[lat, long]:", location)
nearby_venues.head()
```

EXTRACTING NECESSARY COLUMNS:

```
n=nearby_venues.drop(['referralId', 'reasons.count',
'reasons.items', 'venue.id',
       'venue.name',
       'venue.location.labeledLatLngs',
'venue.location.distance',
       'venue.location.cc',
       'venue.categories', 'venue.photos.count',
'venue.photos.groups',
       'venue.location.crossStreet',
'venue.location.address','venue.location.city',
       'venue.location.state', 'venue.location.crossStreet',

'venue.location.neighborhood',  'venue.venuePage.id',

'venue.location.postalCode','venue.location.country'],axis=1)
n.columns
n
```

	venue.location.lat	venue.location.lng	venue.location.formattedAddress	restaurant	others
0	17.423817	78.487257	[Opposite Hussain Sagar Lake (Tank Bund Road),...	6	9
1	17.487673	78.542793	[Sainikpuri, Andhra Pradesh, India]	5	1
2	17.440081	78.484293	[sindhi colony (sindhi colony road), Hyderabad...	34	17
3	17.404284	78.481458	[Himayathnagar, Hyderabad 500029, Telangana, I...	28	11
4	17.400678	78.488575	[Himayatnagar (Narayanguda-himayat Nagar X Roa...	24	13
...	...	...	...	...	...
95	17.368964	78.517230	[Hyderabad, Telangana, India]	6	4
96	17.396277	78.425094	[Toli Chowli (M), Hyderabad 500058, Telangana,...	16	1
97	17.438342	78.395606	[Plot 66, Jyothi Celeste, Kavuri Hills, Madhap...	40	16
98	17.457379	78.363723	[Kondapur, Hyderabad 500084, TG, India]	29	12
99	17.452630	78.363296	[Level 10, SLN Terminus, Hyderabad 500081, Tel...	22	12

DROPPING NAN VALUES:

```
n= n.dropna()
n = n.rename(columns={'venue.location.lat': 'lat',
'venue.location.lng': 'long'})
n
```

CONVERT EVERY ROW OF COLUMN ‘venue.location.formattedAddress’ FROM LIST TO STRING:

```
n['venue.location.formattedAddress']
```

```
0     [Opposite Hussain Sagar Lake (Tank Bund Road),...
1                   [Sainikpuri, Andhra Pradesh, India]
2     [sindhi colony (sindhi colony road), Hyderabad...
3     [Himayathnagar, Hyderabad 500029, Telangana, I...
4     [Himayatnagar (Narayanguda-himayat Nagar X Roa...
                            ...
95                       [Hyderabad, Telangana, India]
96    [Toli Chowli (M), Hyderabad 500058, Telangana,...
97    [Plot 66, Jyothi Celeste, Kavuri Hills, Madhap...
98              [Kondapur, Hyderabad 500084, TG, India]
99    [Level 10, SLN Terminus, Hyderabad 500081, Tel...
Name: venue.location.formattedAddress, Length: 100, dtype: object
```

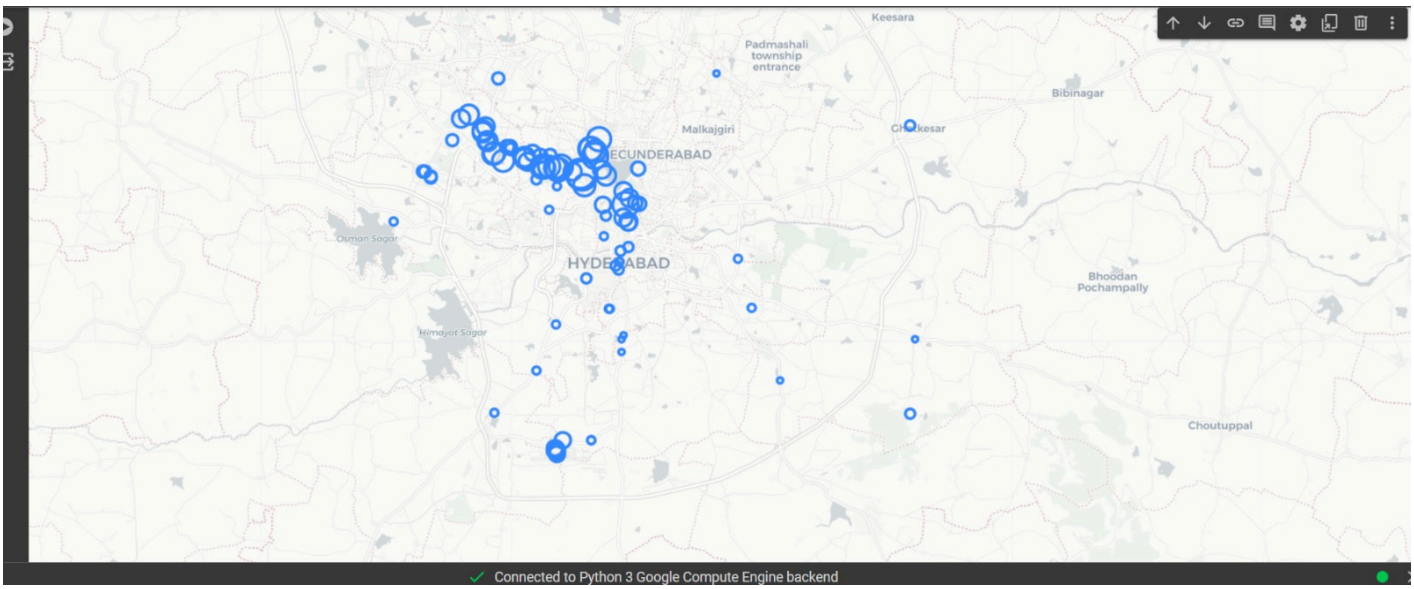
```
spec_chars = ["[","]"]
for char in spec_chars:
  n['venue.location.formattedAddress'] =
n['venue.location.formattedAddress'].astype(str).str.replace(
char, ' ')
n
```

	lat	long	venue.location.formattedAddress	restaurant	others
0	17.423817	78.487257	'Opposite Hussain Sagar Lake (Tank Bund Road)...	6	9
1	17.487673	78.542793	'Sainikpuri', 'Andhra Pradesh', 'India'	5	1
2	17.440081	78.484293	'sindhi colony (sindhi colony road)', 'Hydera...	34	17
3	17.404284	78.481458	'Himayathnagar', 'Hyderabad 500029', 'Telanga...	28	11
4	17.400678	78.488575	'Himayatnagar (Narayanguda-himayat Nagar X Ro...	24	13
...	...	...	...	...	...
95	17.368964	78.517230	'Hyderabad', 'Telangana', 'India'	6	4
96	17.396277	78.425094	'Toli Chowli (M)', 'Hyderabad 500058', 'Telan...	16	1
97	17.438342	78.395606	'Plot 66, Jyothi Celeste, Kavuri Hills, Madha...	40	16
98	17.457379	78.363723	'Kondapur', 'Hyderabad 500084', 'TG', 'India'	29	12
99	17.452630	78.363296	'Level 10, SLN Terminus', 'Hyderabad 500081',...	22	12

PLOT THE CLUSTERED LOCATIONS ON MAP:

```
x, y = "lat", "long"
color = "restaurant"
```

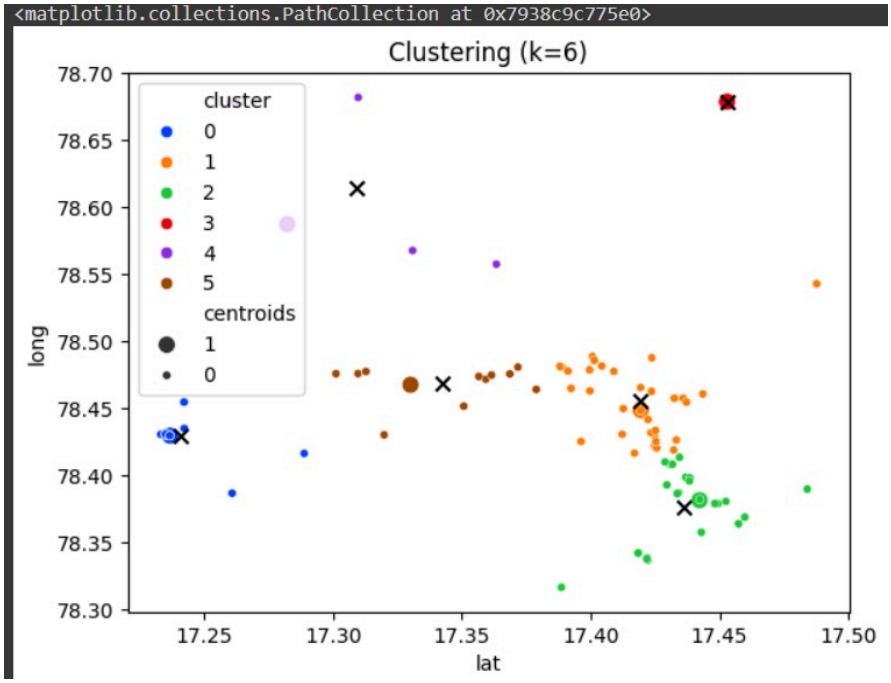
```
size = "others"
popup = "venue.location.formattedAddress"
data = n.copy()
## create color column
lst_colors=["red","green","orange"]
lst_elements = sorted(list(n[color].unique()))
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)
## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]],popup=row[popup],
           radius=row["size"]).add_to(map_), axis=1)
## add html legend

## plot the map
map_
```



K-MEANS CLUSTERING VISUALIZATION:

```
## plot
fig, ax = plt.subplots()
sns.scatterplot(x="lat", y="long", data=n,
```

```
                palette=sns.color_palette("bright",k),
                hue='cluster', size="centroids",
size_order=[1,0],
                legend="brief", ax=ax).set_title('Clustering
(k='+str(k)+')')
th_centroids = model.cluster_centers_
ax.scatter(th_centroids[:,0], th_centroids[:,1], s=50,
c='black',
           marker="x")
```

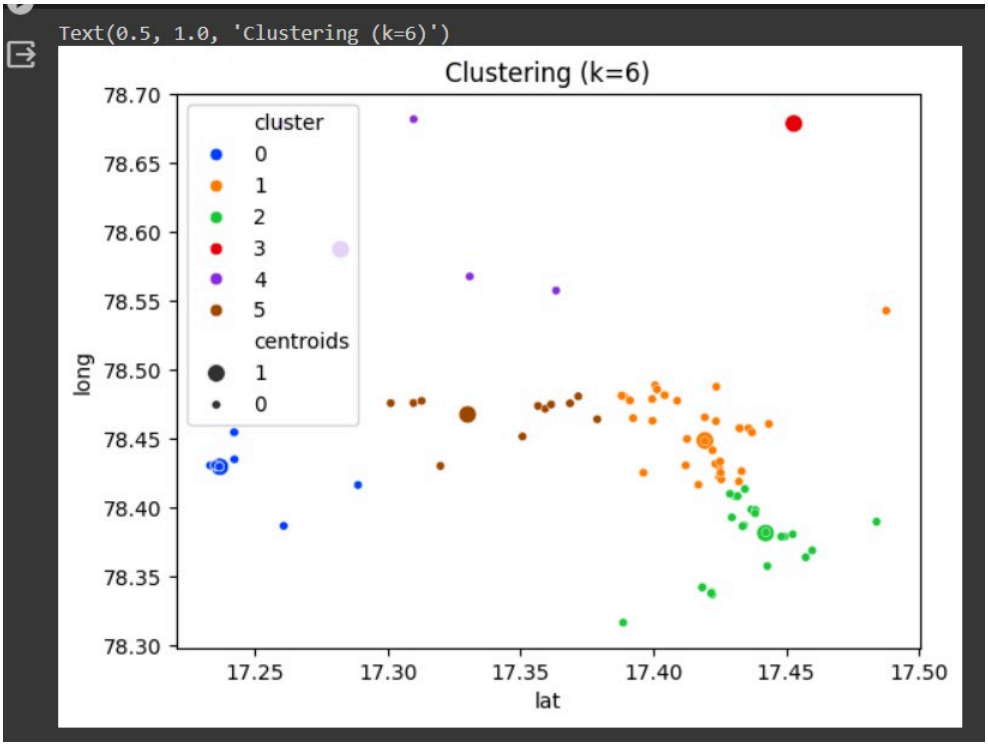


AFFINITY PROPAGATION VISUALIZATION:

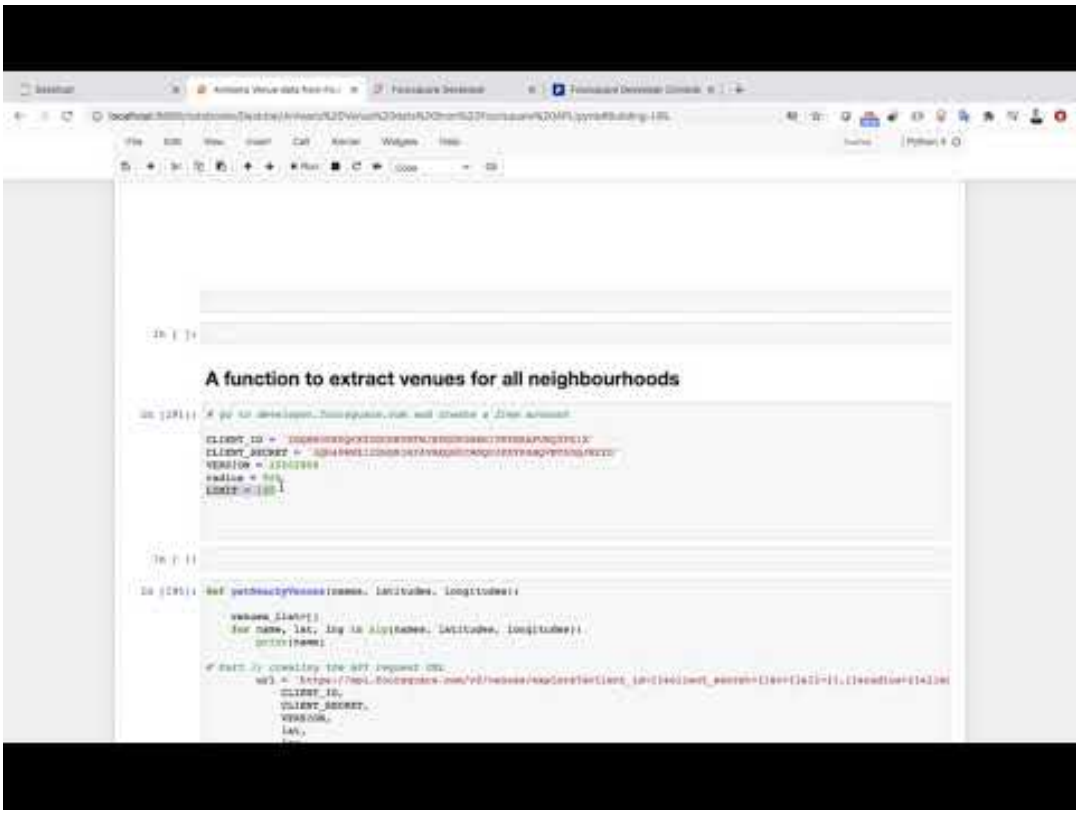
```
model = cluster.AffinityPropagation()
k = n["cluster"].nunique()
sns.scatterplot(x="lat", y="long", data=n,
                palette=sns.color_palette("bright",k),
                hue='cluster', size="centroids",
size_order=[1,0],
                legend="brief").set_title('Clustering
(k='+str(k)+')')
```

FINAL VISUALIZATION:

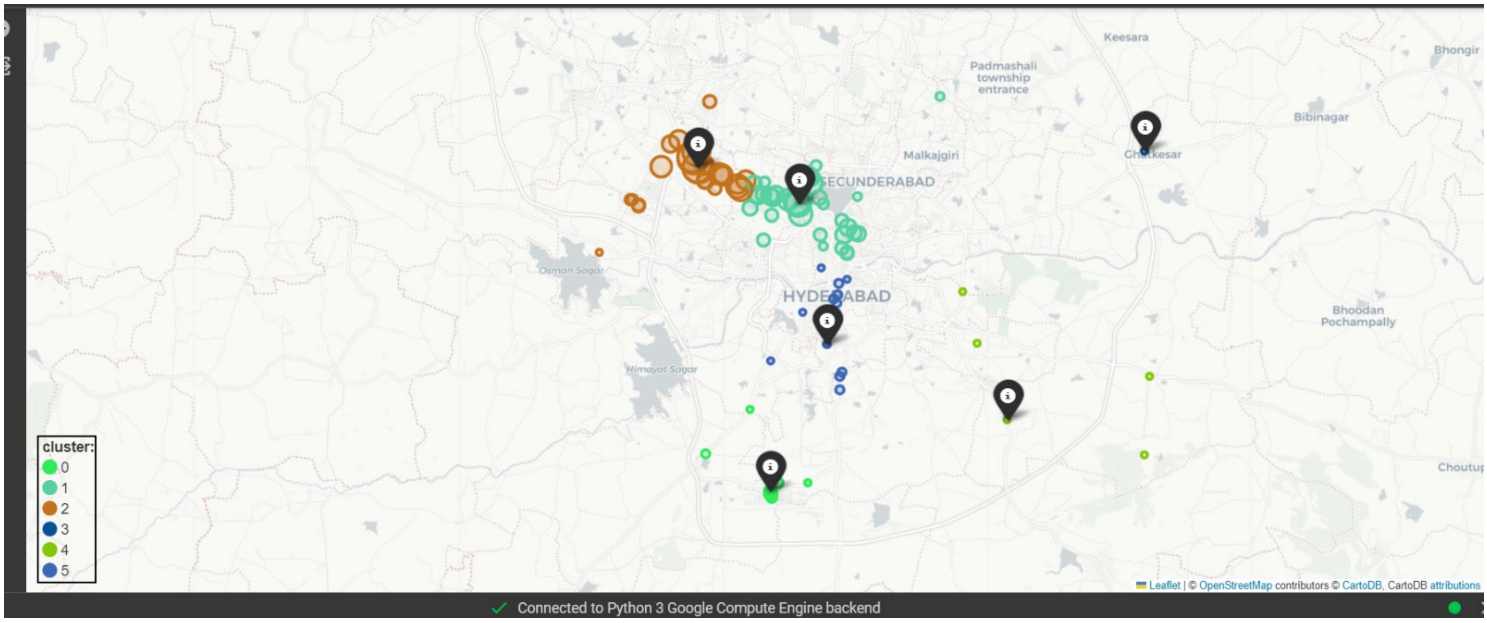
```
x, y = "lat", "long"
color = "cluster"
size = "restaurant"
```




```
popup = "venue.location.formattedAddress"
marker = "centroids"
data = n.copy()
## create color column
lst_elements = sorted(list(n[color].unique()))
```



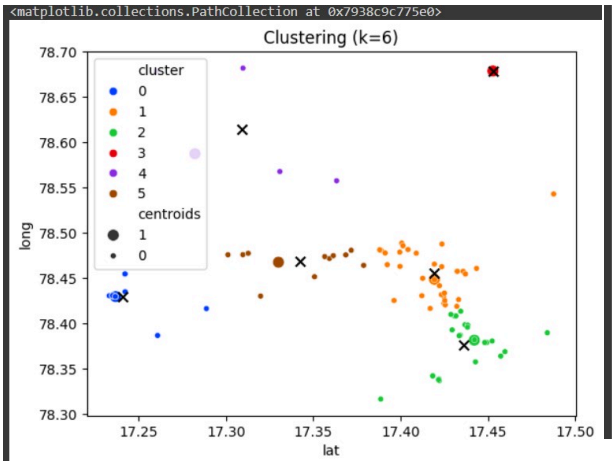
```
lst_colors = ['#%06X' % np.random.randint(0, 0xFFFFFF) for i
in
               range(len(lst_elements))]
data["color"] = data[color].apply(lambda x:
                 lst_colors[lst_elements.index(x)])
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
                data[size].values.reshape(-1,1)).reshape(-1)
## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]],
           color=row["color"], fill=True,popup=row[popup],
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """ """+color+""":"""
for i in lst_elements:
     legend_html = legend_html+"""
      """+str(i)+"""</br>"""
legend_html = legend_html+"""</br></br>"""
map_.get_root().html.add_child(folium.Element(legend_html))
## add centroids marker
lst_elements = sorted(list(n[marker].unique()))
data[data[marker]==1].apply(lambda row:
           folium.Marker(location=[row[x],row[y]],
           draggable=False,  popup=row[popup] ,
           icon=folium.Icon(color="black")).add_to(map_),
axis=1)
## plot the map
map_
```

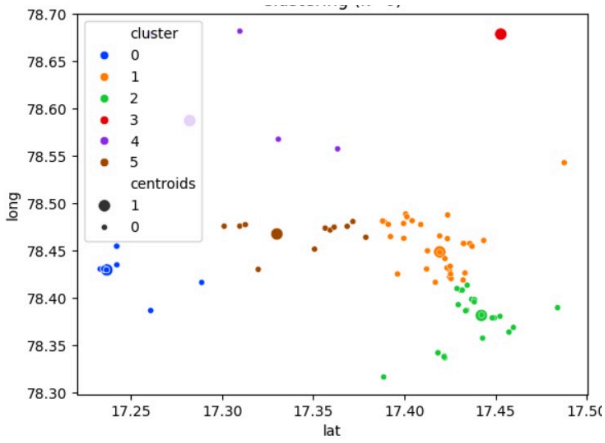


RESULT ANALYSIS:

COMPARISON OF TWO CLASSIFIERS:

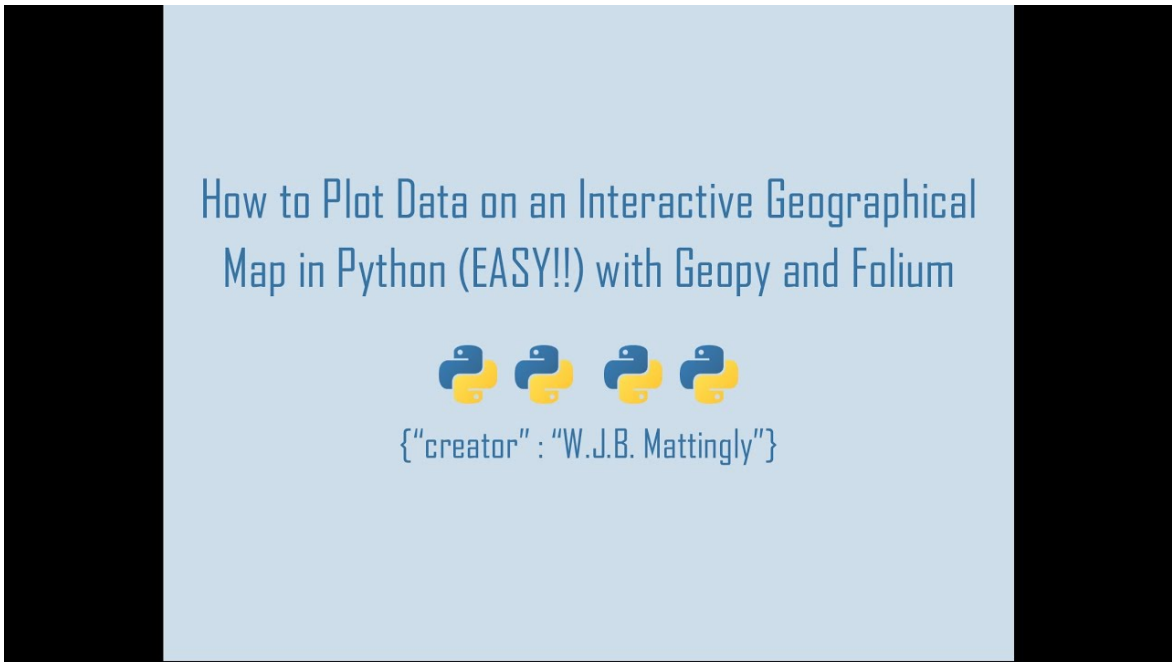
- K-MEANS CLUSTERING require/need predefined centroids which can be specified based on domain knowledge or through techniques like the elbow method.
- We estimated the number of clusters to be 6 using the elbow method.
- AFFINITY PROPAGATION is comparatively better as we don't have to specify the number of clusters. This makes exploratory analysis much easier.





REFERENCES:

1. Venue data extraction using Foursquare API :
2. Using geopy and folium to visualise map:



3. Clustering algorithms: https://repositorio.usp.br/directbitstream/91efa03d-6840-42d8-

a711-cbb8dd3458e0/PROD028505_2922598.pdf

4. K-means clustering: https://ieeexplore.ieee.org/document/5453745

5. Affinity propagation: https://letsdatascience.com/affinity-propagation-clustering/

6. Scikit-learn: https://scikit-learn.org/stable/modules/clustering.html

7. Project plan: file:///Users/meena/Desktop/STUDY/Geolocational%20Data%20Analysis.pdf

8. https://www.ijeast.com/papers/84-87,%20Tesma0711,IJEAST.pdf

9. https://www.irjet.net/archives/V9/i7/ICIETET22/IRJET-V9I701.pdf

10. https://github.com/cntejas/Exploratory-Analysis-Of-Geolocational-Data/blob/main/README.md

CONCLUSION:

Exploring data with pairplot and boxplot visualizations reveals correlations and patterns. K-means clustering identifies clusters in demographic and geolocational data, optimized using the Elbow Method. Insights from geolocational analysis inform urban planning, marketing, and tourism decisions. Future directions include exploring alternative clustering algorithms and integrating additional features for deeper insights. Advanced techniques like geospatial analysis and deep learning offer further potential for enhancing data understanding.