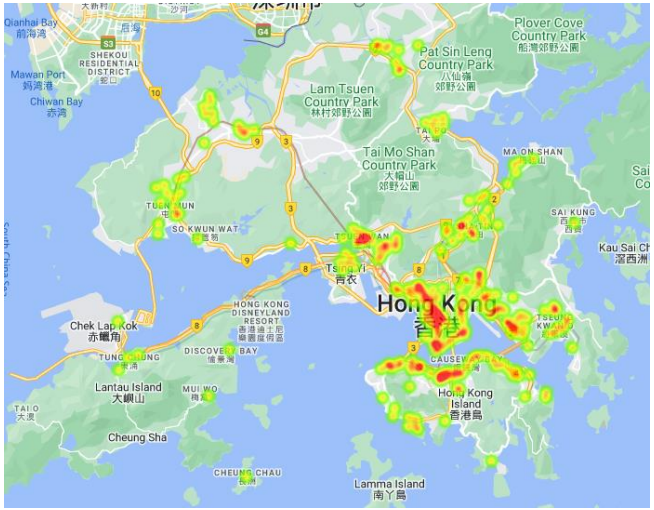
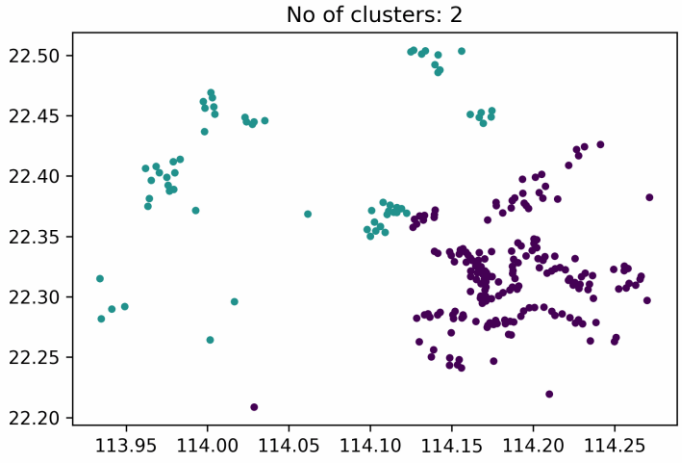
VISUALIZATION & CLUSTERING OF GEOLOCATION DATA

Sibo Ding

Nov 28, 2022





METHODS

01

Convert address to coordinates

googlemaps

02

Plot heatmap

gmaps

03

Cluster coordinates

sklearn.cluster

matplotlib.pyplot

04

Animate clusters (GIF)

matplotlib.

animation

DATA SOURCE

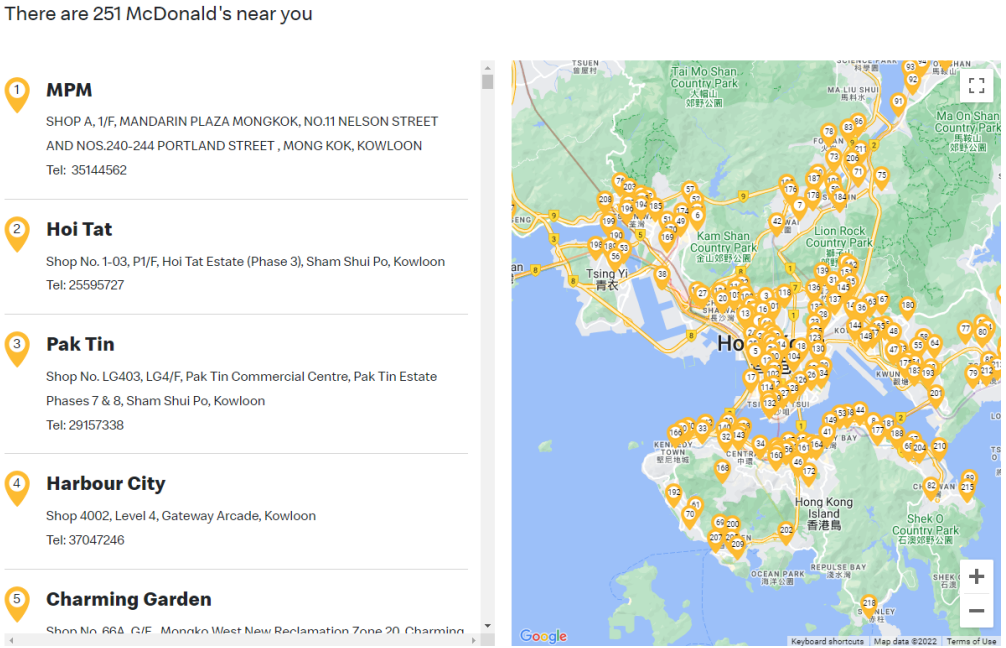
Companies' websites in HK

McDonald's
https://www.mcdonalds.com.hk/en/find-a-restaurant/

Fairwood
https://www.fairwood.com.hk/en/stores

Café de Coral
https://www.eatcdc.com/eng/main/terms.jsp?id=B2D0P2M0R0Y891E8L1D18188M4G0A802

Starbucks
https://www.starbucks.com.hk/en/store-locator/#86



CONVERSION TO COORDINATES AND CLEANING

Import data frame

Add latitude and longitude of each address into two lists

Add two "lat" and "lng" columns; export data frame

Select an address in data frame

Convert the address to latitude and longitude using Google API

"g": a list containing multi-level dictionaries and lists, with composite geolocation info

```
import pandas as pd
import googlemaps

def add_to_coor(csv_in, csv_out):
    df = pd.read_csv(csv_in)

    list_lat = []
    list_lng = []
    for i in range(len(df)):
        g = gmaps_key.geocode(df['address'][i])
        lat = g[0]['geometry']['location']['lat']
        lng = g[0]['geometry']['location']['lng']
        list_lat.append(lat)
        list_lng.append(lng)

    df['lat'] = list_lat
    df['lng'] = list_lng
    df.to_csv(csv_out, index=False)
```

CONVERSION TO COORDINATES AND CLEANING

- Problem 1: Some addresses are unrecognized
  - Narrow the scope and find the unrecognized one, manually add/change/delete street, district info, etc.

- Problem 2: Some addresses are not in HK (Aberdeen in the UK)
  - Find min and max and compare with the coordinates of HK

Output data frame

	A	B	C	D
1	name	address	lat	lng
2	MPM	SHOP A, 1,	22.3186	114.169
3	Hoi Tat	Shop No. 1	22.3289	114.152
4	Pak Tin	Shop No. L	22.3365	114.167
5	Harbour C	Shop 4002	22.2988	114.168
6	Charming	Charming	22.3142	114.165

HEATMAP

```
import gmaps
from ipywidgets.embed import embed_minimal_html

def heatmap(csv_in, html_out):
    df = pd.read_csv(csv_in)
    locations = df[['lat', 'lng']]

    fig = gmaps.figure()
    heatmap_layer = gmaps.heatmap_layer(locations)
    heatmap_layer.opacity = 0.75
    fig.add_layer(heatmap_layer)

    embed_minimal_html(html_out, views=[fig])
```

Filter "lat" and "lng" columns

Create heatmap layer

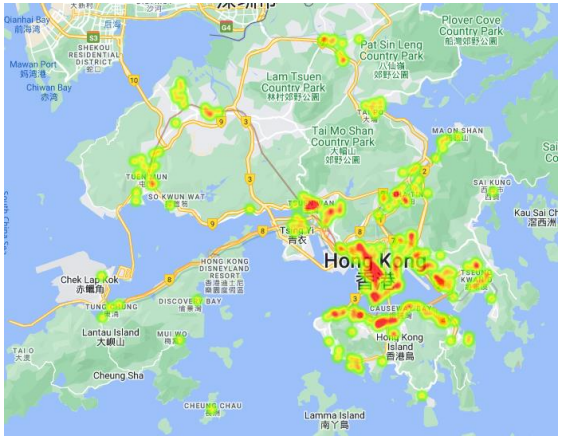
Adjust the opacity of heat

Export .html file

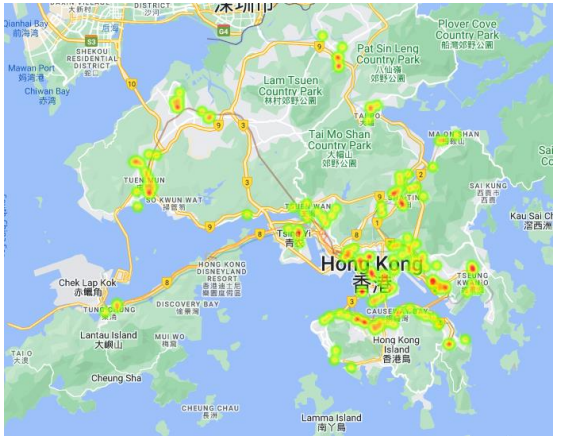
Create an empty Google map figure

Add heatmap layer

HEATMAP

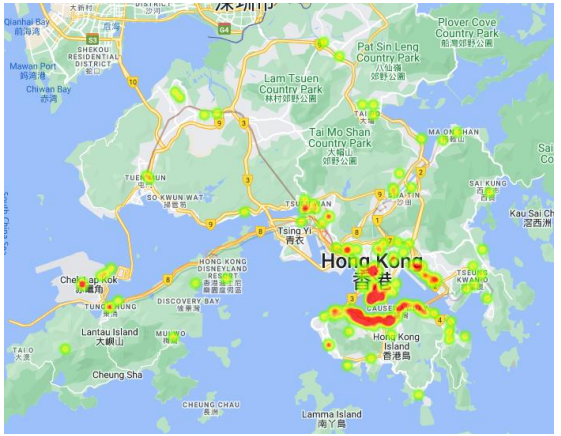


McDonald's



Fairwood

Café de Coral



Starbucks

K-MEANS CLUSTERING

- Unsupervised learning (algorithm learns patterns from untagged data)
- Partition *n* observations into *k* clusters (*k* is determined by human)
- Each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid)
- Inertia: Within-cluster sum-of-squares, measuring how internally coherent clusters are

K-MEANS CLUSTERING

```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

def k_means(csv_in, elbow_clusters, plot_clusters):
    df = pd.read_csv(csv_in)

    x = df['lng'].to_list()
    y = df['lat'].to_list()
    coor = [*zip(x, y)]

    inertias = []
    for i in range(1, elbow_clusters+1):
        kmeans = KMeans(n_clusters=i)
        kmeans.fit(coor)
        inertias.append(kmeans.inertia_)

    plt.plot(range(1, elbow_clusters+1), inertias,
     marker='o')
    plt.title('Elbow method')
    plt.xlabel('Number of clusters')
    plt.ylabel('Inertia')
    plt.show()

    kmeans = KMeans(n_clusters=plot_clusters)
    kmeans.fit(coor)

    plt.scatter(x, y, c=kmeans.labels_)
    plt.title(f'No. of clusters: {plot_clusters}')
    plt.show()
```

Convert "lng" and "lat" columns to lists; combine them into a tuple

Train K-means model

Plot different clusters; “c” means different colors

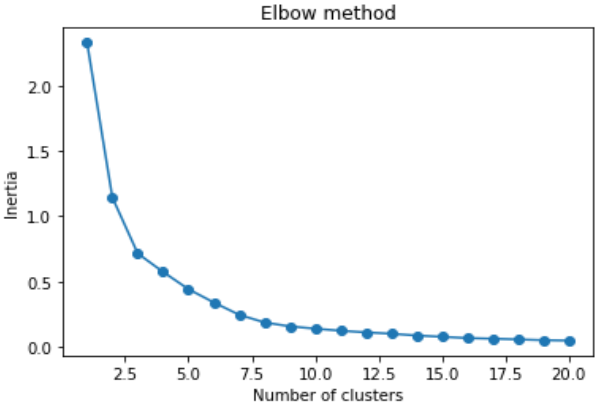
This part generates "elbow" graph, can be omitted to speed up.

Add the inertia of each cluster number to a list

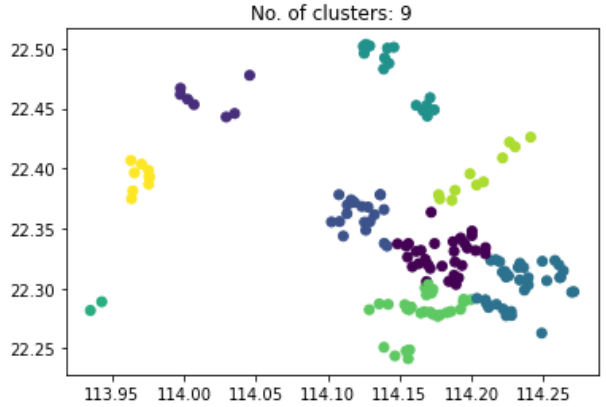
Plot the inertia list

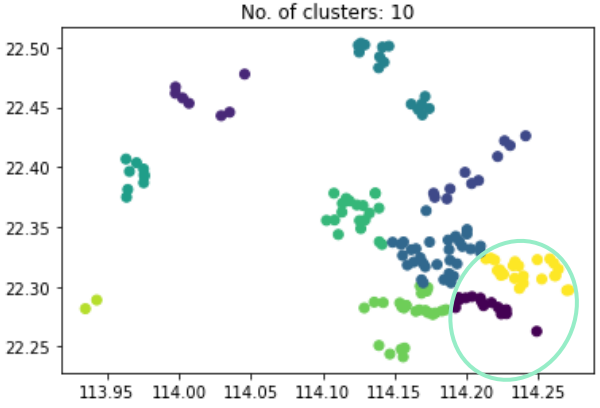
kmeans.lables_: An array of numbers ranging (0, num_clusters - 1), indicating different clusters

K-MEANS CLUSTERING: MCDONALD'S

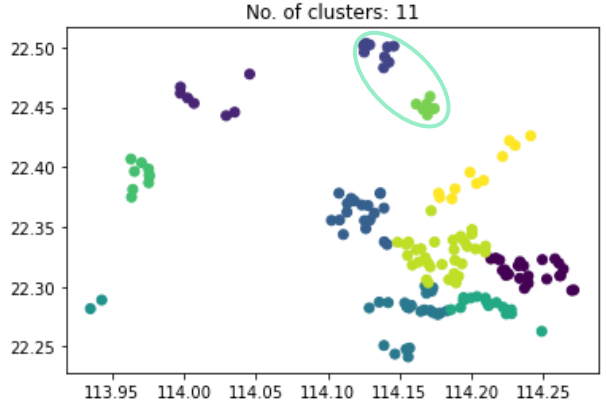


Look for the "elbow" (where the inertia becomes more linear)



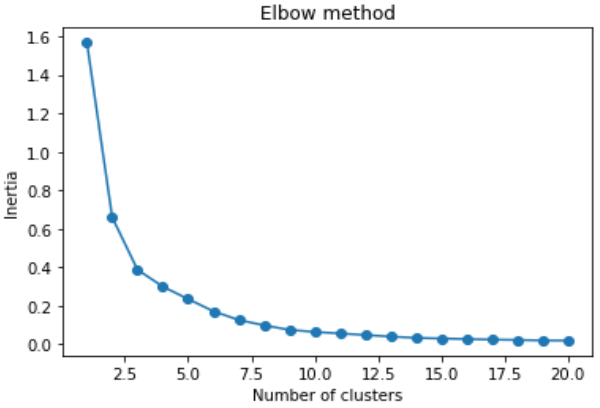


Split Eastern District and Tseung Kwan O

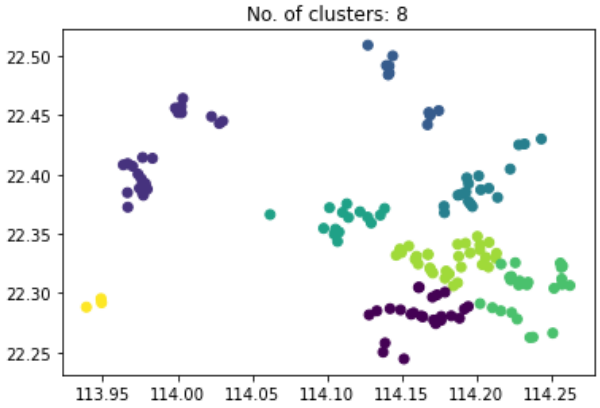


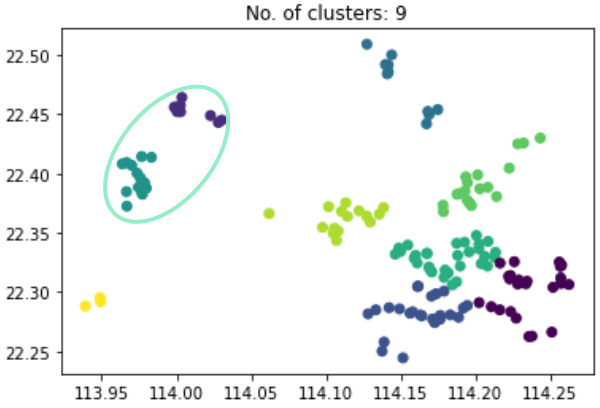
Split Sheung Shui and Tai Po

K-MEANS CLUSTERING: FAIRWOOD

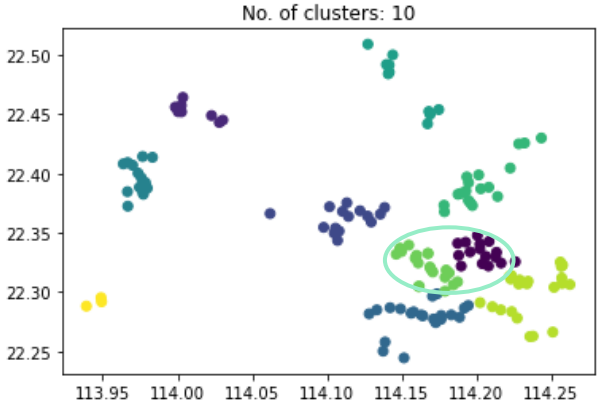


Look for the "elbow" (where the inertia becomes more linear)



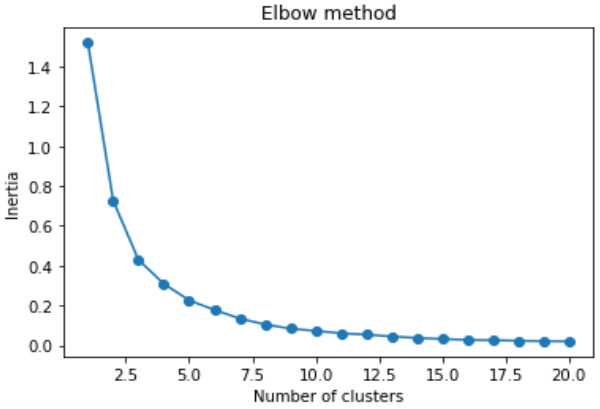


Split Tuen Mun and Yuen Long

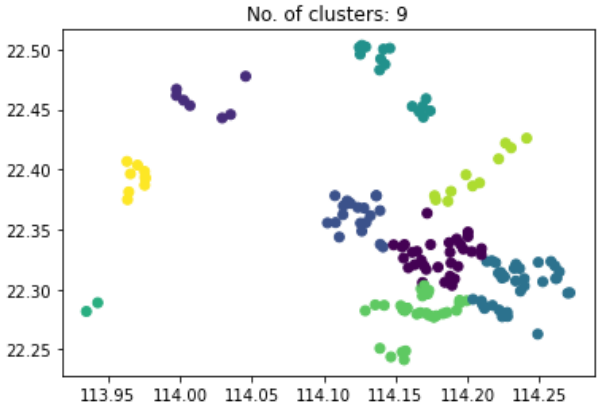


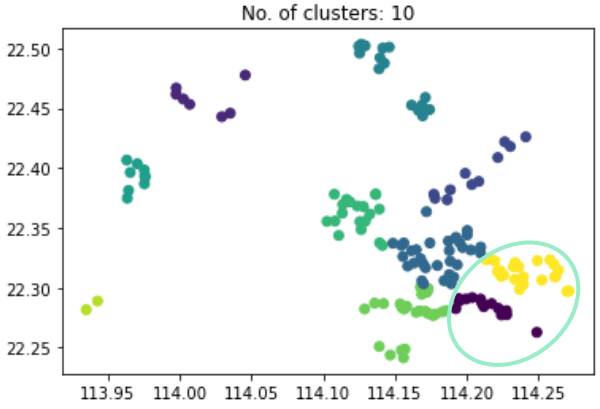
Split West Kowloon and Middle Kowloon

K-MEANS CLUSTERING: CAFÉ DE CORAL

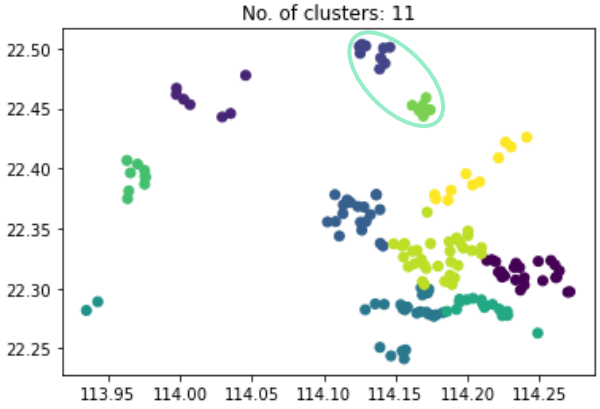


Look for the "elbow" (where the inertia becomes more linear)



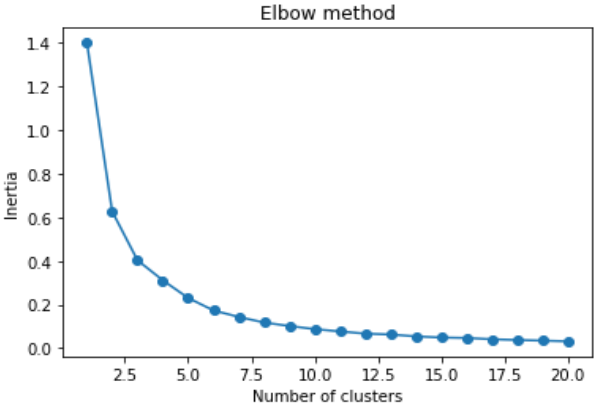


Split Eastern District and Tseung Kwan O

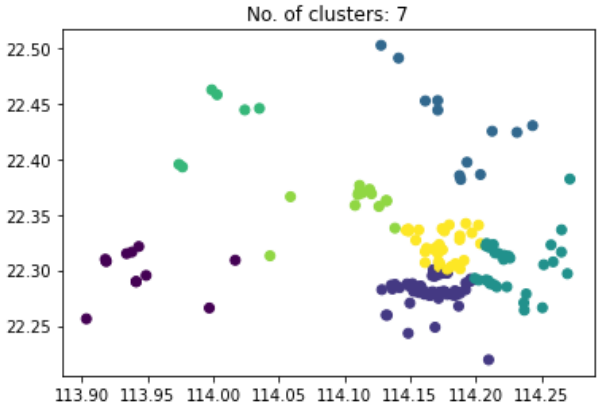


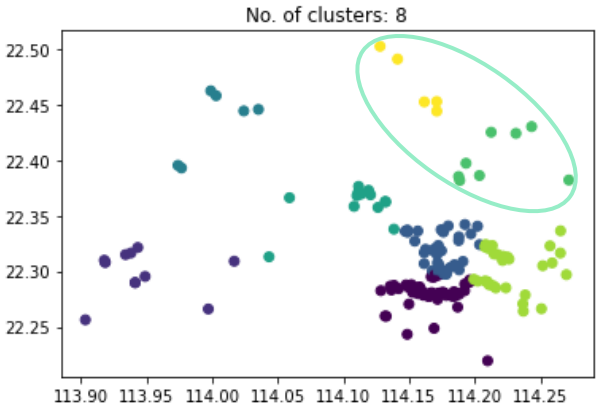
Split Sheung Shui and Tai Po

K-MEANS CLUSTERING: STARBUCKS

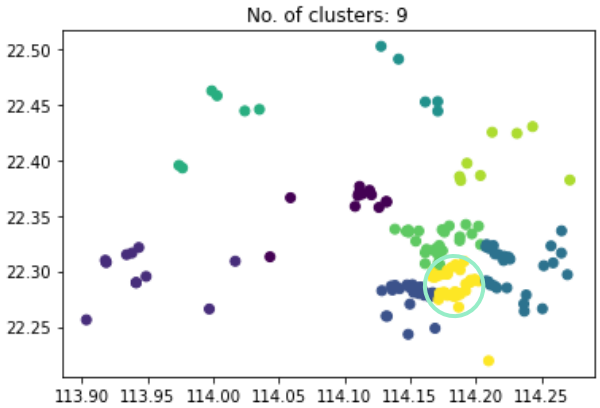


Look for the "elbow" (where the inertia becomes more linear)





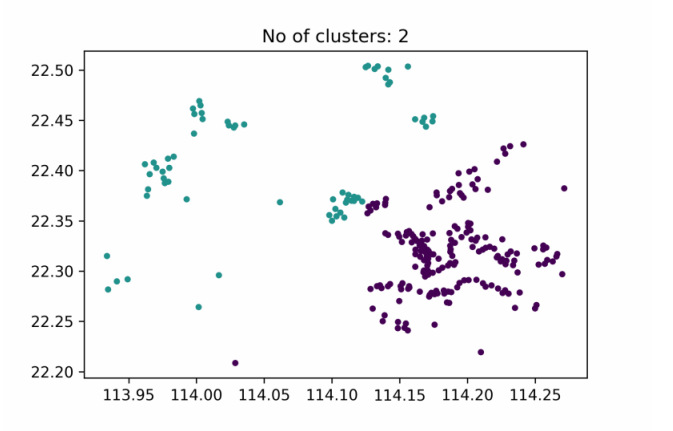
Split Northern District and Ma On Shan, Sai Kung



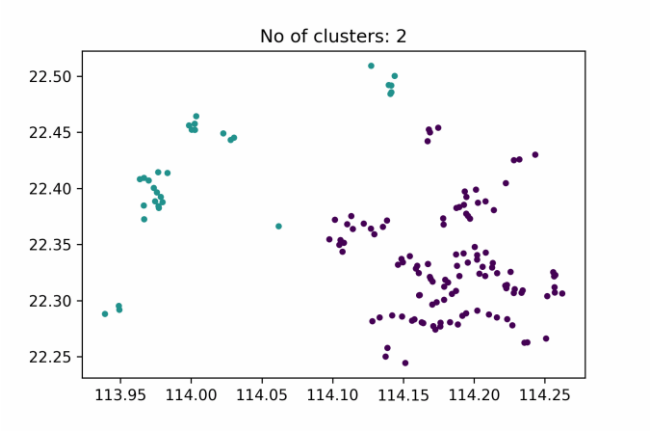
Create Wan Chai and Tsim Sha Tsui

CLUSTER ANIMATION

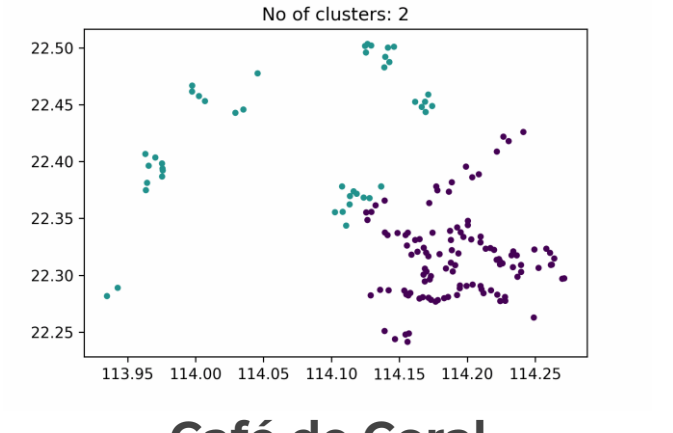
To see dynamic changes with different numbers of clusters



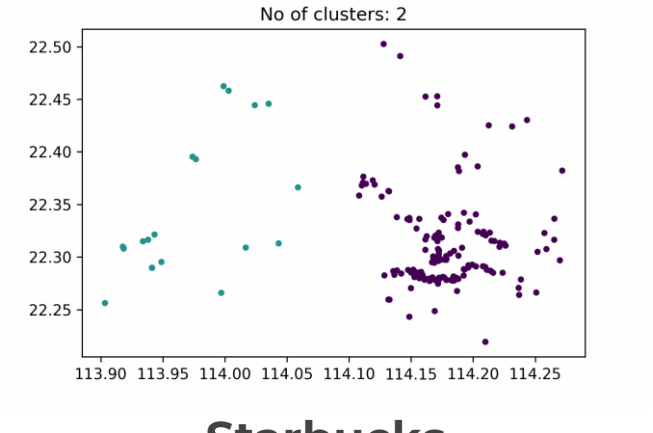
McDonald's



Fairwood



Café de Coral



Starbucks

CLUSTER ANIMATION

```
from matplotlib.animation import FuncAnimation, PillowWriter

fig, ax = plt.subplots()
kmeans = KMeans(n_clusters=2)
kmeans.fit(coor)
scat = plt.scatter(x, y, c=kmeans.labels_, s=10)

max_clusters = 15
color_data = np.array([])
for cluster in range(2, max_clusters+1):
    kmeans = KMeans(n_clusters=cluster)
    kmeans.fit(coor)
    color_data = np.append(color_data, kmeans.labels_/cluster)
color_data = color_data.reshape(max_clusters-1, len(coor))
```

Create a figure with one subplot

Create the initial scatter plot (2 clusters)

Train K-means model

Append kmeans labels of each cluster number to "color_data" array; /cluster: Convert data to range [0,1)

Reshape:
No. of clusters *
No. of coordinates

CLUSTER ANIMATION (CONT.)

```
def update_plot(i, data, scat):
    scat.set_array(data[i])
    ax.set_title(f'No of clusters: {i+2}')
    return scat

ani = FuncAnimation(fig, update_plot, frames=range(max_clusters-1),
                    fargs=(color_data, scat))
ani.save('sb.gif', dpi=300, writer=PillowWriter(fps=0.5))
```

Update scatter plots after the initial plot;
Update colors;
Update title

mpl animation

Export .gif file;
dpi: dots per inch;
fps: frames per second

FURTHER RESEARCH

- Extend to more chain restaurants (Michelin, Tam Jai, Tai Hing, etc.)
- Extend to other industries (HSBC, Wellcome, Centaline Property, etc.)
- Use machine learning to predict/select new location
- Calculate the distance (to nearest MTR, market, etc.)
- Combine with other sociodemographic/geographic layers

REFERENCE

Geocoding

https://www.natasshaselvaraj.com/a-step-by-step-guide-on-geocoding-in-python/
https://developers.google.com/maps/documentation/geocoding/overview

Heatmap

https://www.storybench.org/how-to-build-a-heatmap-in-python/
https://jupyter-gmaps.readthedocs.io/en/latest/tutorial.html
https://jupyter-gmaps.readthedocs.io/en/latest/export.html

K-means Clustering

https://www.w3schools.com/python/python_ml_k-means.asp#:~:text=K%2Dmeans%20is%20an%20unsupervised,the%20variance%20in%20each%20cluster.
https://medium.com/chung-yi/ml%E5%85%A5%E9%96%80-%E4%BA%8C%E5%8D%81%E4%B8%80-knn%E8%88%87k-means%E5%B7%AE%E7%95%B0-7dc6ad0227fc
https://scikit-learn.org/stable/modules/clustering.html#k-means
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.scatter.html

Matplotlib Animation

https://stackoverflow.com/questions/9401658/how-to-animate-a-scatter-plot
https://matplotlib.org/stable/api/_as_gen/matplotlib.animation.FuncAnimation.html
https://eli.thegreenplace.net/2016/drawing-animated-gifs-with-matplotlib/
https://stackoverflow.com/questions/68960005/saving-an-animated-matplotlib-graph-as-a-gif-file-results-in-a-different-looking