Retail Rationalization Business Clustering

Store Rationalization adalah reorganisasi perusahaan untuk meningkatkan efisiensi operasi dan mengurangi biaya. akibat krisis covid-19 memfokuskan investasinya untuk membuat bisnisnya lebih digital.

kita akan menggunakan "set data Starbucks Stores" yang menyediakan lokasi semua toko yang beroperasi. kita akan memilih daerah geografi tertentu dan, selain garis lintang dan garis bukur yang disediakan, kita akan mensimulasikan beberapa informasi bisnis untuk setiap toko dalam kumpulan data (biaya, kapasitas, staf).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import folium
import geopy

from sklearn import preprocessing, cluster
import scipy
```

```
df = pd.read_csv('https://raw.githubusercontent.com/mdipietro09/DataScience_ArtificialIntelligence_Utils/master/machine_learning/data_stores.csv')
df.head()
```

	Brand	Store Number	Store Name	Ownership Type	Street Address	City	State/Province	Country	Postcode	Phone Number	Timezone	Longitude	Latitude
0	Starbucks	47370-257954	Meritxell, 96	Licensed	Av. Meritxell, 96	Andorra la Vella	7	AD	AD500	376818720	GMT+1:00 Europe/Andorra	1.53	42.51
1	Starbucks	22331-212325	Ajman Drive Thru	Licensed	1 Street 69, Al Jarf	Ajman	AJ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.42
2	Starbucks	47089-256771	Dana Mall	Licensed	Sheikh Khalifa Bin Zayed St.	Ajman	AJ	AE	NaN	NaN	GMT+04:00 Asia/Dubai	55.47	25.39

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25600 entries, 0 to 25599
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Brand           25600 non-null  object
```

```
 1   Store Number    25600 non-null  object
 2   Store Name      25600 non-null  object
 3   Ownership Type  25600 non-null  object
 4   Street Address  25598 non-null  object
 5   City            25585 non-null  object
 6   State/Province  25600 non-null  object
 7   Country         25600 non-null  object
 8   Postcode        24078 non-null  object
 9   Phone Number    18739 non-null  object
 10  Timezone        25600 non-null  object
 11  Longitude       25599 non-null  float64
 12  Latitude        25599 non-null  float64
dtypes: float64(2), object(11)
memory usage: 2.5+ MB
```

```
df.City.value_counts().count()
```

```
5469
```

Dataset diatas memiliki lebih dari 25.000 toko dan 5.469 kota, tapi untuk studi case ini kita hanya akan menggunakan satu kota

```
dtf = df[df['City'] == 'Las Vegas'][['City','Street Address','Longitude','Latitude']].reset_index(drop=True)
dtf = dtf.reset_index().rename(columns={'index':'id'})
dtf.head()
```

	id	City	Street Address	Longitude	Latitude
0	0	Las Vegas	4507 Flamingo Rd	-115.20	36.12
1	1	Las Vegas	475 E Windmill Lane, Fashion Show	-115.15	36.04
2	2	Las Vegas	3200 LAS VEGAS BLVD. S., STE 1795	-115.17	36.13
3	3	Las Vegas	8350 W Cheyenne Ave	-115.28	36.22

```
dtf.count()
```

```
id                156
City              156
Street Address    156
Longitude         156
Latitude          156
dtype: int64
```

kawasan 'Las Vegas' terdapat 156 toko. untuk melanjutkan kasus bisnis, kita akan mensimulasikan beberapa informasi untuk setiap toko

- Potensi: total kapasitas dalam hal staf(misalnya 10 berarti toko dapat memiliki hingga 10 karyawan)
- Staf: tingkat staf saat ini (misalnya 7 berarti tokok saat ini beroperasi dengan 7 staf)
- Kapasitas: kapasitas kiri saat ini (misalnya 10-7=3, toko masih dapat menampung 3 karyawan
- Biaya: biaya tahunan bagi perusahaan untuk menjaga agar toko tetap beroperasi ('rendah','sedang','tinggi')

```
dtf['Potential'] = np.random.randint(low=3, high=10+1, size=len(dtf))
dtf['Staff'] = dtf['Potential'].apply(lambda x: int(np.random.rand()*x)+1)
dtf['Capasity'] = dtf['Potential'] - dtf['Staff']
dtf['Cost'] = np.random.choice(['Hight','Medium','Low'], size=len(dtf), p=[0.4, 0.5, 0.1])
dtf.head()
```

	id	City	Street Address	Longitude	Latitude	Potential	Staff	Capasity	Cost
0	0	Las Vegas	4507 Flamingo Rd	-115.20	36.12	6	5	1	Medium
1	1	Las Vegas	475 E Windmill Lane, Fashion Show	-115.15	36.04	9	9	0	Hight
2	2	Las Vegas	3200 LAS VEGAS BLVD. S., STE 1795	-115.17	36.13	8	1	7	Medium
3	3	Las Vegas	8350 W Cheyenne Ave	-115.28	36.22	9	8	1	Medium
4	4	Las Vegas	3730 LAS VEGAS BLVD S	-115.18	36.11	7	6	1	Hight

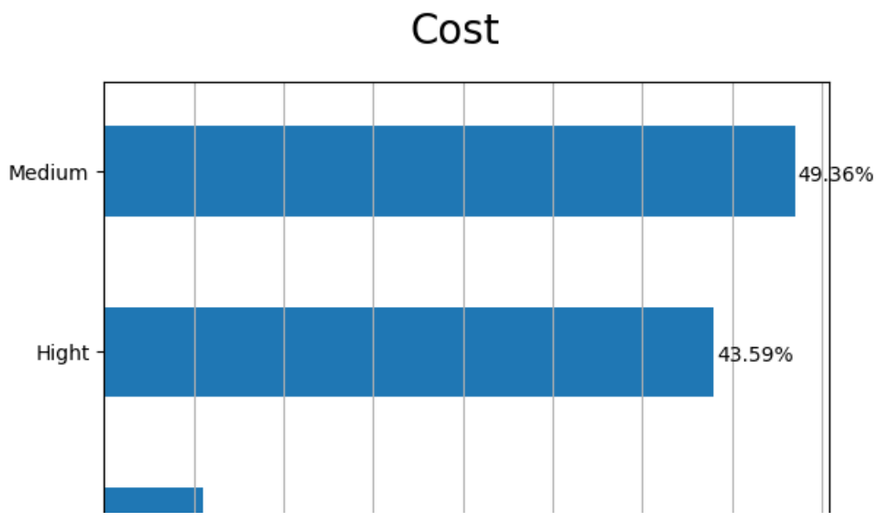
angka-angka diatas yang baru saja dibuat adalah simulasi data

EDA

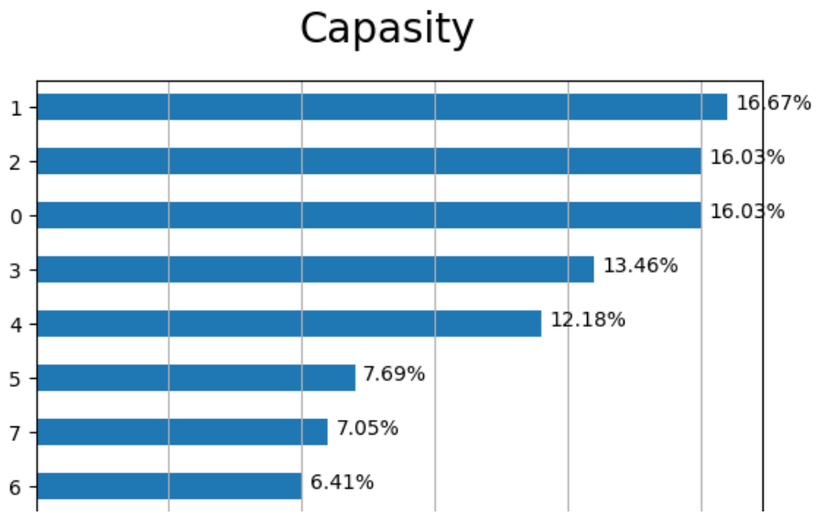
Anggap saja kita meiliki bisnis ritel dan kita harus menutup bebrapa toko. kita ingin melakukan itu dengan memaksimalkan keuntungan (dengan meminimalkan biaya) dan tanpa memberhentikan staff

```
x='Cost'
ax = dtf[x].value_counts().sort_values().plot(kind='barh')

totals = []

for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)
for i in ax.patches:
    ax.text(i.get_width()+.3, i.get_y()+.20,
            str(round((i.get_width()/total)*100, 2))+'%',
            fontsize=10, color='black')
ax.grid(axis="x")
```

```
plt.suptitle(x, fontsize=20)
plt.show()
```




```
ax = dtf['Capasity'].value_counts().sort_values().plot(kind='barh')
totals = []

for i in ax.patches:
    totals.append(i.get_width())
total = sum(totals)

for i in ax.patches:
    ax.text(i.get_width() + .3, i.get_y()+.20,
            str(round((i.get_width()/total) * 100, 2)) + '%',
            fontsize=10, color='black')
ax.grid(axis='x')
plt.suptitle('Capasity', fontsize=20)
plt.show()
```



saat ini, hanya sebagian kecil toko yang beroperasi dengan potensi penuh (kapasitas kiri = 0), artinya ada beberapa dengan staff yang sangat rendah

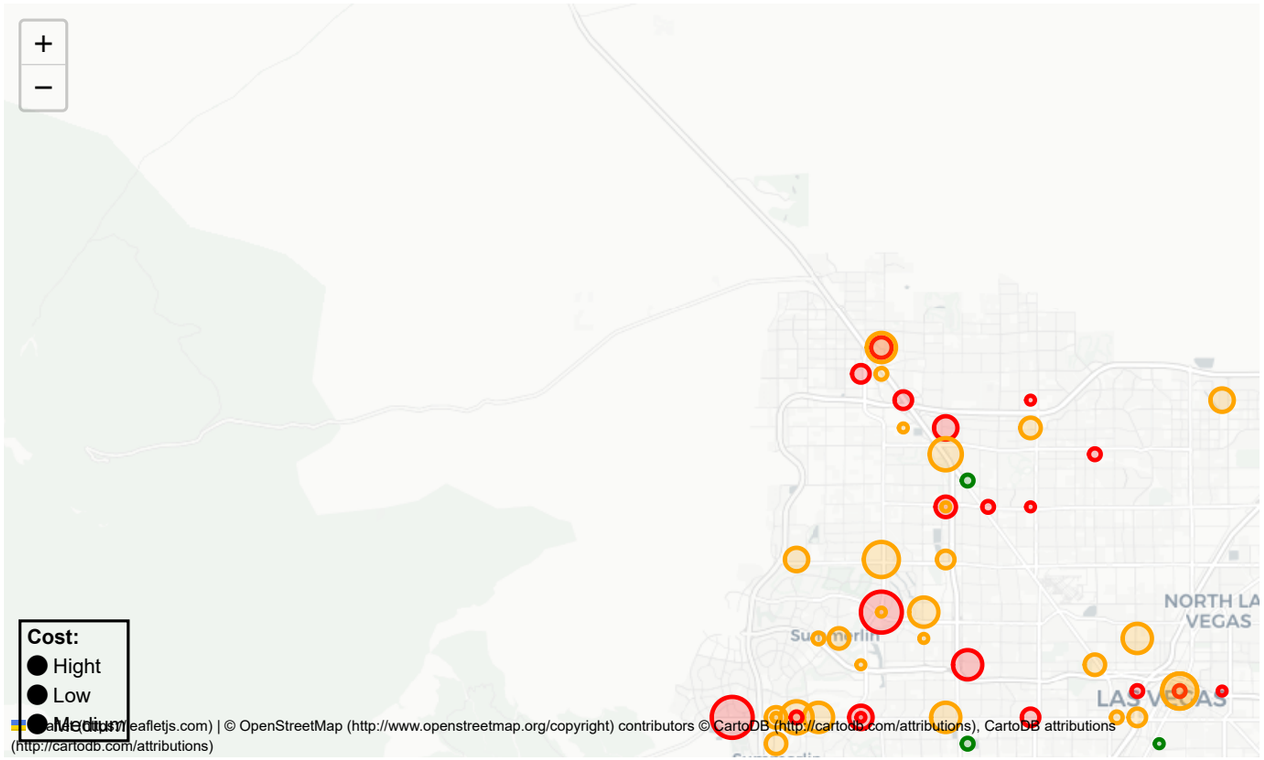
Visualisasi Geografi

```
city = 'Las Vegas'
# 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)
```

```
Las Vegas, Clark County, Nevada, United States
[lat, long]: [36.1672559, -115.148516]
```

```
x, y = "Latitude", "Longitude"
color = "Cost"
size = "Staff"
popup = "Street Address"
data = dtf.copy()
```

```
## create color column
lst_colors=["red","green","orange"]
lst_elements = sorted(list(dtf[color].unique()))
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]], popup=row[popup],
           color=row["color"], fill=True,
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:2px solid black; z-index:9999; font-size:14px;"> <b>"""+color+""":</b><br>"""
for i in lst_elements:
     legend_html = legend_html+""" <i class="fa fa-circle
     fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+""""">
     </i> """+str(i)+"""<br>"""
legend_html = legend_html+"""</div>"""
map_.get_root().html.add_child(folium.Element(legend_html))

## plot the map
map_
```



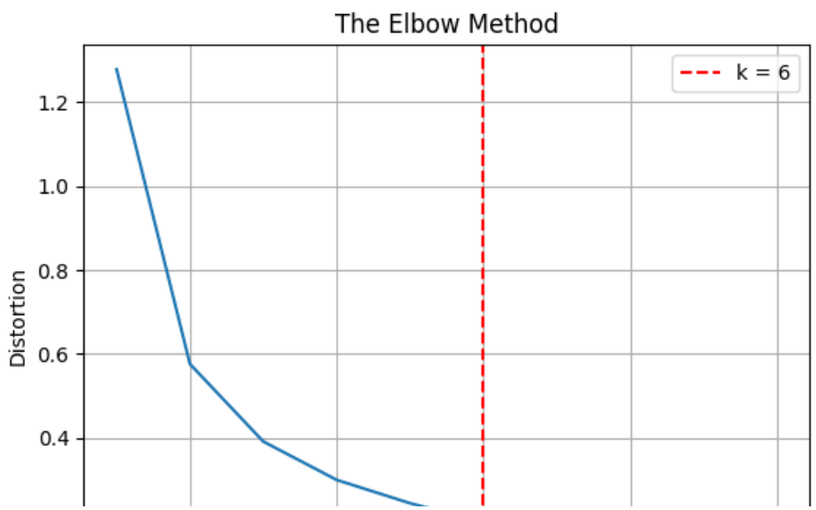
tujuan kita adalah untuk menutup toko berbiaya tinggi(titik merah) dengan memindahkan staff ke toko biaya rendah, dengan kapasitas yang terletak di lingkungan yang sama. akibatnya, kita akan memaksimalkan keuntungan dan efisiensi

Evaluasi dan modeling

K-means

untuk menemukan K yang tepat, kita akan menggunakan metode elbow: memplot varians sebagai fungsi dari jumlah cluster dan memilih k yang meratakan kurva

```
x = dtf[['Latitude','Longitude']]
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()
```




```
k = 5
```

```
model = cluster.KMeans(n_clusters=k, init='k-means++')
X = dtf[["Latitude","Longitude"]]
## clustering
dtf_X = X.copy()
dtf_X["cluster"] = model.fit_predict(X)
## find real centroids
closest, distances = scipy.cluster.vq.vq(model.cluster_centers_,
                     dtf_X.drop("cluster", axis=1).values)
dtf_X["centroids"] = 0
for i in closest:
    dtf_X["centroids"].iloc[i] = 1
## add clustering info to the original dataset
dtf[["cluster","centroids"]] = dtf_X[["cluster","centroids"]]
dtf.sample(5)
```

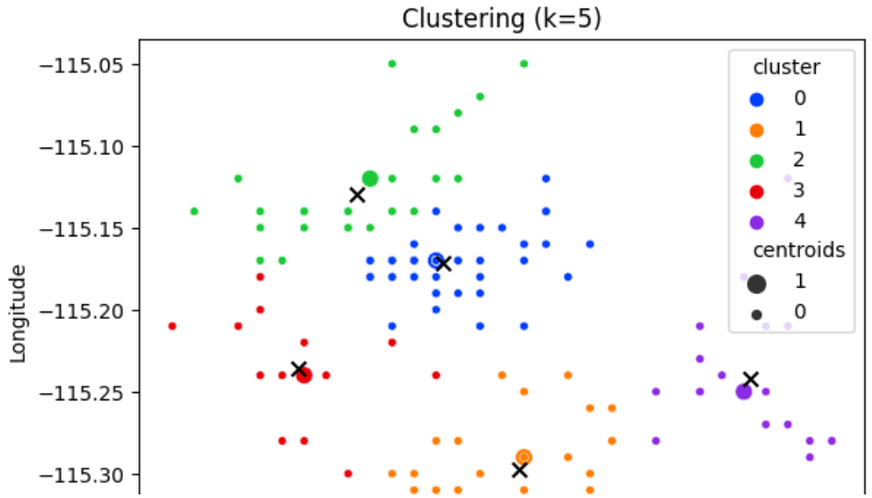
```
/usr/local/lib/python3.9/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
<ipython-input-25-5f4f897a64d5>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
  dtf_X["centroids"].iloc[i] = 1
```

	id	City	Street Address	Longitude	Latitude	Potential	Staff	Capasity	Cost	cluster	centroids
146	146	Las Vegas	10430 West Cheyenne Ave, Lynden Square	-115.32	36.22	7	5	2	Medium	1	0
			7400 S. Las Vegas								

Kita menambahkan dua kolom ke dataset: "cluster" yang menunjukkan cluster apa yang menjadi milik observasi, dan "centroids" yaitu 1 jika pengamatan juga merupakan centroid (paling dekat dengan pusat) dan 0 sebaliknya. Mari kita rencanakan:

```
## plot
fig, ax = plt.subplots()
sns.scatterplot(x="Latitude", y="Longitude", data=dtf,
                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")
```

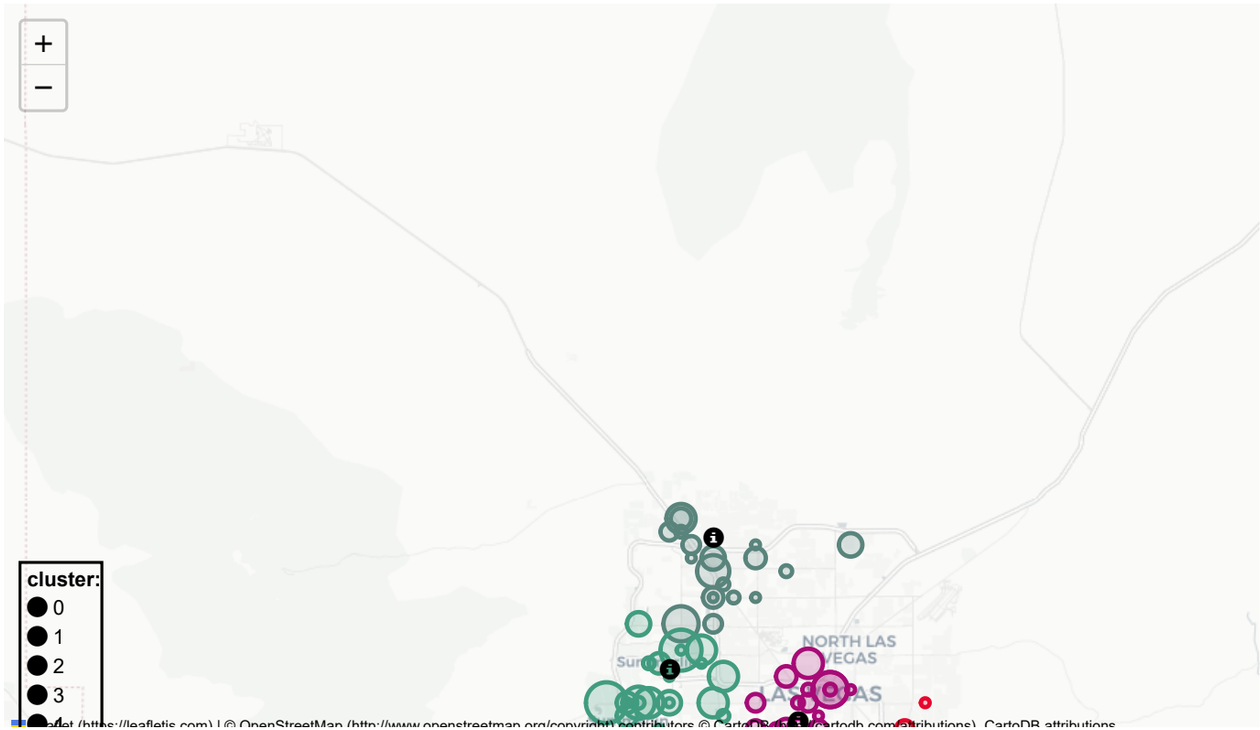
```
<matplotlib.collections.PathCollection at 0x7fcdf1e7d670>
```



Terlepas dari algoritma yang Anda gunakan untuk mengelompokkan data, sekarang Anda memiliki kumpulan data dengan dua kolom lagi ("cluster", "centroids"). Kita dapat menggunakannya untuk memvisualisasikan cluster pada peta, dan kali ini kita akan menampilkan centroid juga menggunakan marker.

```
x, y = "Latitude", "Longitude"
color = "cluster"
size = "Staff"
popup = "Street Address"
marker = "centroids"
data = dtf.copy()
## create color column
lst_elements = sorted(list(dtf[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]], popup=row[popup],
           color=row["color"], fill=True,
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:2px solid black; z-index:9999; font-size:14px;"> <b>"""+color+""":</b><br>"""
for i in lst_elements:
     legend_html = legend_html+""" <i class="fa fa-circle
     fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+""""">
     </i> """+str(i)+"""<br>"""
legend_html = legend_html+"""</div>"""
map_.get_root().html.add_child(folium.Element(legend_html))
## add centroids marker
lst_elements = sorted(list(dtf[marker].unique()))
data[data[marker]==1].apply(lambda row:
           folium.Marker(location=[row[x],row[y]],
           popup=row[marker], draggable=False,
           icon=folium.Icon(color="black")).add_to(map_), axis=1)
## plot the map
map_
```



Sekarang kita memiliki cluster, kita dapat memulai rasionalisasi toko di dalam masing-masing cluster.

Store Rationalization

Karena fokus utama artikel ini adalah mengelompokkan data geospasial, Kita akan membuat bagian ini tetap sederhana. Di dalam setiap cluster, Kita akan memilih target potensial (toko berbiaya tinggi) dan hub (toko berbiaya rendah), dan merelokasi staf target di hub sampai yang terakhir mencapai kapasitas penuh. Ketika seluruh staf target dipindahkan, toko bisa ditutup.

```
dtf_new = pd.DataFrame()
for c in sorted(dtf["cluster"].unique()):
    dtf_cluster = dtf[dtf["cluster"]==c]

    ## hubs and targets
    lst_hubs = dtf_cluster[dtf_cluster["Cost"]=="low"
               ].sort_values("Capacity").to_dict("records")
    lst_targets = dtf_cluster[dtf_cluster["Cost"]=="high"
               ].sort_values("Staff").to_dict("records")
    ## move targets
```

```
        for target in lst_targets:
             for hub in lst_hubs:
                 ### if hub has space
                 if hub["Capacity"] > 0:
                    residuals = hub["Capacity"] - target["Staff"]
                    #### case of hub has still capacity: do next target
                    if residuals >= 0:
                       hub["Staff"] += target["Staff"]
                       hub["Capacity"] = hub["Potential"] - hub["Staff"]
                       target["Capacity"] = target["Potential"]
                       target["Staff"] = 0
                       break
                    #### case of hub is full: do next hub
                    else:
                       hub["Capacity"] = 0
                       hub["Staff"] = hub["Potential"]
                       target["Staff"] = -residuals
                       target["Capacity"] = target["Potential"] - target["Staff"]
    dtf_new = dtf_new.append(pd.DataFrame(lst_hubs)
                  ).append(pd.DataFrame(lst_targets))
dtf_new = dtf_new.append(dtf[dtf["Cost"]=="medium"]
                  ).reset_index(drop=True).sort_values(
                  ["cluster","Staff"])
dtf_new.head()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-28-2db66c8f4f6a> in <cell line: 2>()
      4
      5     ## hubs and targets
----> 6     lst_hubs = dtf_cluster[dtf_cluster["Cost"]=="low"
      7                 ].sort_values("Capacity").to_dict("records")
      8     lst_targets = dtf_cluster[dtf_cluster["Cost"]=="high"

                                    2 frames
/usr/local/lib/python3.9/dist-packages/pandas/core/generic.py in _get_label_or_level_values(self, key, axis)
   1848             )
   1849         else:
-> 1850             raise KeyError(key)
   1851
   1852         # Check for duplicates
```

Ini adalah algoritma yang sangat sederhana yang dapat ditingkatkan dalam beberapa cara: misalnya, dengan memasukkan toko-toko berbiaya menengah ke dalam persamaan dan meniru prosesnya ketika toko-toko berbiaya rendah semuanya penuh.

Mari kita lihat berapa banyak toko mahal yang kita tutup dengan proses dasar ini:

```
dtf_new["closed"] = dtf_new["Staff"].apply(lambda x: 1
                                           if x==0 else 0)
print("closed:", dtf_new["closed"].sum())


x, y = "Latitude", "Longitude"
color = "cluster"
size = "Staff"
popup = "Street Address"
marker = "centroids"
data = dtf_new.copy()
## create color column
lst_elements = sorted(list(dtf[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]], popup=row[popup],
           color=row["color"], fill=True,
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:2px solid black; z-index:9999; font-size:14px;"> <b>"""+color+""":</b><br>"""
for i in lst_elements:
     legend_html = legend_html+""" <i class="fa fa-circle
     fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+""";">
     </i> """+str(i)+"""<br>"""
legend_html = legend_html+"""</div>"""
map_.get_root().html.add_child(folium.Element(legend_html))
## add centroids marker
lst_elements = sorted(list(dtf[marker].unique()))
data[data[marker]==1].apply(lambda row:
           folium.Marker(location=[row[x],row[y]],
           popup=row[marker], draggable=False,
           icon=folium.Icon(color="black")).add_to(map_), axis=1)
## plot the map
map_
```

0s completed at 3:58 PM