ЗАВДАННЯ Т9 «КЛАСТЕРИЗАЦІЯ»

Уявіть, що міжнародне круїзне агентство Carnival Cruise Line вирішило себе розреклямувати за допомогою банерів і звернулося для цього до вас. Щоб протестувати, чи велика від таких банерів користь, їх буде розміщено всього 20 штук по усьому світі. **Вам треба вибрати 20 таких локацій для розміщення, щоб користь була великою і агентство продовжило з вами співробітничати.**

Агентство велике, і в нього є кілька офісів по усьому світі. Поблизу цих офісів воно й хоче розмістити банери — легше домовлятися про установку банерів та перевіряти результат. Також ці місця повинні бути популярні серед туристів.

Для пошуку оптимальних місць скористаємося базою даних найбільшої соціальної мережі, заснованої на локаціях — **Foursquare**.

Частина відкритих даних є, наприклад, на сайті **archive.org**: https://archive.org/details/201309_foursquare_dataset_umn

Скачаємо архів **fsq.zip** із цієї сторінки.

Для зручної роботи із цим документом перетворимо його до формату csv, видаливши рядки, які не мають координат – вони неінформативні для нас.

Завантажуємо необхідні бібліотеки.

In [1]:
```
import numpy as np
import pandas as pd
import csv
from sklearn.cluster import MeanShift
```

In [2]:
```
with open('checkins.dat') as input_file:
    newLines = []
    for line in input_file:
        newLine = [x.strip() for x in line.split('|')]
        if len(newLine) == 5 and newLine[3] and newLine[4]:
            newLines.append(newLine)
```

In [3]:
```
with open('checkins.csv', 'w') as output_file:
    file_writer = csv.writer(output_file)
    file_writer.writerows(newLines)
```

In [4]:
```
data = pd.read_csv('checkins.csv', header=0)
```

За допомогою **pandas** побудуємо DataFrame і переконаємося, що всі **396634** рядка з координатами зчитані з файлу успішно.

In [5]:
```
data.shape
```

Out[5]:
```
(396634, 6)
```

Тепер необхідно **кластеризувати координати**, щоб виявити центри скупчень туристів. Оскільки банери мають порівняно невелику площу дії, нам потрібний **алгоритм, що дозволяє обмежити розмір кластера і щоб він не залежав від кількості кластерів.**

Ця задача — гарний привід познайомитися з **алгоритмом MeanShift**, про який ми не обговорювали у лекції. Його опис при бажанні можна подивитися в sklearn user guide.

Використайте MeanShift, вказавши bandwidth=0.1, що в перекладі із градусів у метри коливається приблизно від 5 до 10 км у середніх широтах.

Примітка: на 396634 рядках кластеризація буде працювати довго. Бути дуже терпляче не забороняється — результат від цього тільки покращиться. Але для того, **щоб здати завдання, знадобиться сабсет з перших 100 тисяч рядків.** Це компроміс між якістю й витраченим часом. Робота алгоритму на усьому датасете займає біля години, а на 100 тис. рядків - приблизно 2 хвилини, однак цього досить для одержання коректних результатів.

Деякі із кластерів, що отримали, **містять занадто мало точок (туристів) - такі кластери не цікаві рекламодавцям. Тому треба визначити, які із кластерів містять, скажемо, більше 15 елементів. Центри цих кластерів і є оптимальними для розміщення банерів.**

При бажанні, щоб побачити отримані результати на карті можна передати центри знайдених кластерів в один з інструментів візуалізації. Наприклад, сайт mapcustomizer.com має функцію **Bulk Entry**, куди можна вставити центри отриманих кластерів у форматі: 38.8951118, -77.0363658

Як ми пам'ятаємо, 20 банерів потрібно розмістити біля офісів компанії. Знайдемо на Google Maps за запитом *Carnival Cruise Line* локації всіх офісів:

- 33.751277, -118.188740 (Los Angeles)
- 25.867736, -80.324116 (Miami)
- 51.503016, -0.075479 (London)
- 52.378894, 4.885084 (Amsterdam)
- 39.366487, 117.036146 (Beijing)
- -33.868457, 151.205134 (Sydney)

Залишилося визначити 20 найближчих до них центрів кластерів. Тобто обчислити відстань до найближчого офісу для кожної точки й вибрати 20 з найменшим значенням.

Примітка: під час обчислення відстаней і кластеризації можна знехтувати тим, що Земля кругла, тому що в точках, розташованих близько одна до одної похибка мала, а в інших точках значення досить великі.

Для здачі завдання введіть координати 20 центрів кластерів та виберіть із знайдених 20 центрів кластерів той, який є найближчим офісу компанії.

Відповідь у цьому завданні — широта й довгота цього центра кластера.

Завантажте свій нотебук та цю широту та довготу у classroom.

In [6]:
```
import sklearn.cluster as cluster
```

In [7]:
```
data.head()
```

Out[7]:

	id	user_id	venue_id	latitude	longitude	created_at
0	984222	15824	5222	38.895112	-77.036366	2012-04-21 17:43:47
1	984234	44652	5222	33.800745	-84.410520	2012-04-21 17:43:43
2	984291	105054	5222	45.523452	-122.676207	2012-04-21 17:39:22
3	984318	2146539	5222	40.764462	-111.904565	2012-04-21 17:35:46
4	984232	93870	380645	33.448377	-112.074037	2012-04-21 17:38:18

Запишемо у змінну **X** широту (**latitude**) та довготу (**longitude**) перших 100 тисяч рядків.

In [8]:
```
X = data.values[0:100000, 3:5]
print(X.shape)
```

Out[8]:
```
(100000, 2)
```

In [9]:
```
X[:5, :5]
```

Out[9]:
```
array([[38.8951118, -77.0363658],
       [33.800745, -84.41052],
       [45.5234515, -122.6762071],
       [40.764462, -111.904565],
       [33.4483771, -112.0740373]], dtype=object)
```

Використовуємо **MeanShift, вказавши bandwidth=0.1**, що в перекладі із градусів у метри коливається приблизно від 5 до 10 км у середніх широтах.

In [10]:
```
cluster1 = cluster.MeanShift(bandwidth=0.1)
```

In [11]:
```
cluster1.fit(X)
```

Out[11]:
```
MeanShift(bandwidth=0.1)
```

In [12]:
```
labels = cluster1.labels_
cluster_centers = cluster1.cluster_centers_
print('len(labels) =', len(labels))
print('len(cluster_centers) =', len(cluster_centers))

labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)
print('len(labels_unique) =', n_clusters_)
```

```
len(labels) = 100000
len(cluster_centers) = 3231
len(labels_unique) = 3231
```

In [13]:
```
cluster_centers
```

Out[13]:
```
array([[  40.7177164 ,  -73.99183542],
       [  33.44943805, -112.00213969],
       [  33.44638027, -111.90188756],
       ...,
       [ -37.8229826 ,  145.1811902 ],
       [ -41.2924945 ,  174.7732353 ],
       [ -45.0311622 ,  168.6626435 ]])
```

In [14]:
```
labels_unique
```

Out[14]:
```
array([   0,    1,    2, ..., 3228, 3229, 3230], dtype=int64)
```

Деякі із кластерів, що отримали, **містять занадто мало точок (туристів) - такі кластери не цікаві рекламодавцям. Тому треба визначити, які із кластерів містять, скажемо, більше 15 елементів. Центри цих кластерів і є оптимальними для розміщення банерів.**

In [15]:
```
d = {}
for label in labels:
    if (label not in d.keys()):
        d[label] = 1
    else:
        d[label] += 1
```

In [16]:
```
count = 0
for key in d.keys():
    if (d[key] > 15):
        count += 1
print('К-сть кластерів, що містять > 15 елементів =', count)
```

```
К-сть кластерів, що містять > 15 елементів = 592
```

In [17]:
```
clusters_select = np.ndarray(shape=(count,2))
```

In [18]:
```
i = 0
j = 0
while (i < len(cluster_centers)):
    if (d[i] > 15):
        clusters_select[j] = cluster_centers[i]
        j += 1
    i += 1
```

In [19]:
```
print('К-сть обраних кластерів =', len(clusters_select))
```

```
К-сть обраних кластерів = 592
```

In [20]:
```
clusters_select[:5, :5]
```

Out[20]:
```
array([[  40.7177164 ,  -73.99183542],
       [  33.44943805, -112.00213969],
       [  33.44638027, -111.90188756],
       [  41.8782478,  -87.62984336],
       [  37.68868157, -122.40933037]])
```

Запишемо координати всіх офісів.

In [21]:
```
offices = np.ndarray(shape=(6,2))
offices[0] = np.array([33.751277, -118.188740]) # Los Angeles
offices[1] = np.array([25.867736, -80.324116])  # Miami
offices[2] = np.array([51.503016, -0.075479])   # London
offices[3] = np.array([52.378894, 4.885084])    # Amsterdam
offices[4] = np.array([39.366487, 117.036146])  # Beijing
offices[5] = np.array([-33.868457, 151.205134]) # Sydney
```

Розрахуємо відстань згідно з координатами широти та довготи до найближчого офісу для кожної точки. Це буде квадратний корінь з суми квадратів різниці відповідних координат.

In [22]:
```
def distance(x, y):
    return np.sqrt(np.sum((x - y)**2))
```

Приклад розрахунку відстані довільних об'єктів, а саме офісу в Лондоні та одного з кластерів.

In [23]:
```
distance(offices[2], clusters_select[100])
```

Out[23]:
```
119.02383962900103
```

Також відстань можна виразити більш розгорнутою формулою. Значення будуть ідентичні.

def distance(point1, point2): return ((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)**0.5 distance(offices[2], clusters_select[100])

In [24]:
```
answer_index = 0
min_dist = 0
i = 0
while (i < len(clusters_select)):
    distances = [distance(xx, clusters_select[i]) for xx in offices]
    if min_dist == 0:
        min_dist = min(distances)
        answer_index = i
    else:
        if min_dist > min(distances):
            min_dist = min(distances)
            answer_index = i
    i += 1
```

In [25]:
```
print('answer_index = ', answer_index)
print('min_dist = ', min_dist)
```

```
answer_index =  417
min_dist =  0.007834758163107856
```

In [26]:
```
ans = clusters_select[answer_index]
print(round(ans[0],3), round(ans[1],3))
```

```
-33.861 151.205
```