Представим, что международное круизное агентство Carnival Cruise Line решило себя разрекламировать с помощью баннеров и обратилось для этого к вам. Чтобы протестировать, велика ли от таких баннеров польза, их будет размещено всего 20 штук по всему миру. Вам надо выбрать 20 таких локаций для размещения, чтобы польза была большой и агентство продолжило с вами сотрудничать.

Агентство крупное, и у него есть несколько офисов по всему миру. Вблизи этих офисов оно и хочет разместить баннеры — легче договариваться и проверять результат. Также эти места должны быть популярны среди туристов.

Для поиска оптимальных мест воспользуемся базой данных крупнейшей социальной сети, основанной на локациях — Foursquare.

Часть открытых данных есть, например, на сайте archive.org:

<https://archive.org/details/201309_foursquare_dataset_umn>

Скачаем любым удобным образом архив fsq.zip с этой страницы.

Нас будет интересовать файл checkins.dat. Для удобной работы с этим документом преобразуем его к формату csv, удалив строки, не содержащие координат — они неинформативны для нас.

***data = pd.read\_csv('checkins.dat',sep='|').replace(r'^\s+$', np.nan, regex=True).dropna()***

С помощью pandas построим DataFrame и убедимся, что все 396634 строки с координатами считаны успешно.

Теперь необходимо кластеризовать данные координаты, чтобы выявить центры скоплений туристов. Поскольку баннеры имеют сравнительно небольшую площадь действия, нам нужен алгоритм, позволяющий ограничить размер кластера и не зависящий от количества кластеров.

Эта задача — хороший повод познакомиться с алгоритмом MeanShift, который мы обошли стороной в основной части лекций. Используйте MeanShift, указав bandwidth=0.1, что в переводе из градусов в метры колеблется примерно от 5 до 10 км в средних широтах.

**Примечание:** на 396634 строках кластеризация будет работать долго. Быть очень терпеливым не возбраняется — результат от этого только улучшится. Но для того, чтобы сдать задание, понадобится сабсет из первых 100 тысяч строк. Это компромисс между качеством и затраченным временем. Обучение алгоритма на всём датасете занимает около часа, а на 100 тыс. строк — примерно 2 минуты, однако этого достаточно для получения корректных результатов.

Некоторые из получившихся кластеров содержат слишком мало точек — такие кластеры не интересны рекламодателям. Поэтому надо определить, какие из кластеров содержат, скажем, больше 15 элементов. Центры этих кластеров и являются оптимальными для размещения.

***val, counts = np.unique(labels\_, return\_counts=True)***

При желании увидеть получившиеся результаты на карте можно передать центры получившихся кластеров в один из инструментов визуализации. Например, сайт [mapcustomizer.com](https://www.mapcustomizer.com) имеет функцию Bulk Entry, куда можно вставить центры полученных кластеров в формате:

38.8951118, -77.0363658

Как мы помним, 20 баннеров надо разместить близ офисов компании. Найдем на Google Maps по запросу Carnival Cruise Line адреса всех офисов:



offices\_coords = np.array([[33.751277, 25.867736, 51.503016, 52.378894, 39.366487, -33.868457], [-118.188740, -80.324116, -0.075479, 4.885084, 117.036146, 151.205134]])

Осталось определить 20 ближайших к ним центров кластеров. Т.е. посчитать дистанцию до ближайшего офиса для каждой точки и выбрать 20 с наименьшим значением.

**Примечание:** при подсчете расстояний и в кластеризации можно пренебречь тем, что Земля круглая, так как в точках, расположенных близко друг к другу погрешность мала, а в остальных точках значение достаточно велико.

Для сдачи задания выберите из получившихся 20 центров тот, который наименее удален от ближайшего к нему офиса. Ответ в этом задании — широта и долгота этого центра, записанные через пробел.

%pylab inline

import pandas as pd

import sklearn.cluster as sk

import matplotlib.pyplot as plt

from itertools import cycle

dataset=pd.read\_csv('checkins.csv')

newdataset=dataset.head(100000)

X=np.array((newdataset['latitude'],newdataset['longitude']))

X=X.T

ms=sk.MeanShift(bandwidth=0.1)

ms.fit(X)

labels = ms.labels\_

cluster\_centers = ms.cluster\_centers\_

labels\_unique = np.unique(labels)

n\_clusters\_ = len(labels\_unique)

plt.figure(1)

plt.clf()

print labels.shape

colors = cycle('bgrcmykbgrcmykbgrcmykbgrcmyk')

for k, col in zip(range(n\_clusters\_), colors):

my\_members = labels == k

cluster\_center = cluster\_centers[k]

plt.plot(X[my\_members,0],X[my\_members,1],col + '.' )#Пространство признаков с выделенными разными цветами кластерами

#plt.plot(cluster\_center[0], cluster\_center[1], 'o', markerfacecolor=col,

# markeredgecolor='k', markersize=14)#центры кластеров

plt.title('Estimated number of clusters: %d' % n\_clusters\_)

plt.show()