Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: современные языки программирования

Отчет по лабораторной работе №4

Выполнил: студент гр. 910101 Атаев И.М.

Проверила: Василькова А.Н.

Минск 2022

**Цель работы** – приобрести навыки работы с исходными данными, получаемыми из закрытых источников сети Интернет, на примере социальной сети «ВК»; получить навыки работы с алгоритмом кластеризации k-means.

Задание:

1. Зарегистрировать Standalone-приложение приложение для социальной сети «ВК» (см. п.2 https://vk.com/dev/first\_guide)

2. Получить ключ доступа - access\_token для авторизации приложения под вашим аккаунтом и возможности его автономной работы (см. п.3 https://vk.com/dev/first\_guide) .

3. Создать приложение, производящее построение социального графа с уровнем вложенности 2 (т.е. «дерева друзей и их друзей») для своего профиля (свой профиль принять за root – т.е. уровень 0).

4. Произвести сбор информации об аккаунтах полученного графа: возраст, место проживания, статус, количество опубликованных фото, видео, заметок пользователя.

5. Произвести кластеризацию пользователей методом k-средних (k-means) по различным признакам, например:

1. возраст (при k = 3,5,7)

2. активность профиля – количество фото на странице (при k = 3,5,7)

3. активность профиля – количество видео на странице (при k = 3,5,7)

4. активность профиля – количество заметок на странице (при k = 3,5,7)

5. активность владельца – количество групп пользователя, в которых он принимает

участие (при k = 5)

6. Привести статистику по полученным кластерам в виде графиков – среднего, медианного и среднеквадратичного отклонения.

**Листинг кода:**

**Lab4API.py**

import matplotlib

import requests

import networkx as nx

matplotlib.use('TkAgg')

import matplotlib.pyplot as plt

from datetime import date

from sklearn.cluster import KMeans

import numpy as np

token='vk1.a.KVXJR7uPAHnSod1m5R-ngeMHrgG3Pk8QVuuv4DXmXr2XEbwHcTd3cNI5Hn7dTNRhRxxN6jmOwL905RkO8QrvlspdKDK5583j\_VHi0cRAmG1BGT-x1xy1cS57U8Zq7uUfWyJdl2IV3iHHO2kzRBp0HXkBEIHdx84ZbM9skNjFBBqGC7kvFVv3xNOiVIbCcLgC'

def fetch\_request(url, key):

response = requests.get(url).json()

if response.get("error"):

return []

return response["response"][key]

def social\_graph\_friends(user\_id, count):

url = f"https://api.vk.com/method/friends.get?user\_id={user\_id}&count={count}&order=hints&access\_token={token}&v=5.81"

friend\_ids = fetch\_request(url, "items")

graph = {user\_id: friend\_ids}

for friend\_id in friend\_ids:

friends\_url = f"https://api.vk.com/method/friends.get?user\_id={friend\_id}&count={count}&order=hints&access\_token={token}&v=5.81"

friends = fetch\_request(friends\_url, "items")

graph[friend\_id] = friends

g = nx.Graph()

for key, value in graph.items():

g.add\_node(key)

g.add\_nodes\_from(value)

edges = [(key, v) for v in value]

g.add\_edges\_from(edges)

plt.title("Социальный граф друзей VK")

nx.draw(g, node\_size=25)

plt.show()

def get\_value\_in\_json\_by\_key(json, key):

if key in json:

return json[key]

else:

return ""

def build\_age(bdate):

bd = bdate.split(".")

if len(bd) != 3:

return 0

day, month, year = map(int, bd)

today = date.today()

age = today.year - year - ((today.month, today.day) < (month, day))

return age

def build\_place(country, city):

place = ""

if country:

place += country

if city:

place += f", {city}"

return place

def build\_friend(json):

bdate = get\_value\_in\_json\_by\_key(json, "bdate")

status = get\_value\_in\_json\_by\_key(json, "status")

country = get\_value\_in\_json\_by\_key(get\_value\_in\_json\_by\_key(json, "country"), "title")

city = get\_value\_in\_json\_by\_key(get\_value\_in\_json\_by\_key(json, "city"), "title")

count\_photos\_url = f"https://api.vk.com/method/photos.getAll?owner\_id={json['id']}&access\_token={token}&v=5.81"

count\_photos = fetch\_request(count\_photos\_url, "count")

if not count\_photos: count\_photos = 0

count\_video\_url = f"https://api.vk.com/method/video.get?owner\_id={json['id']}&access\_token={token}&v=5.81"

count\_video = fetch\_request(count\_video\_url, "count")

if not count\_video: count\_video = 0

count\_notes\_url = f"https://api.vk.com/method/notes.get?user\_id={json['id']}&access\_token={token}&v=5.81"

count\_notes = fetch\_request(count\_notes\_url, "count")

if not count\_notes: count\_notes = 0

return {"id": json["id"],

"age": build\_age(bdate),

"place": build\_place(country, city),

"status": status,

"count\_photos": count\_photos,

"count\_video": count\_video,

"count\_notes": count\_notes}

def fetch\_friends\_data(user\_id, count):

url = f"https://api.vk.com/method/friends.get?user\_id={user\_id}&fields=status,bdate,country,city&count={count}&order=hints&access\_token={token}&v=5.81"

friends = fetch\_request(url, "items")

friends\_data = []

for friend in friends:

user = build\_friend(friend)

friends\_data.append(user)

print(user)

friends\_url = f"https://api.vk.com/method/friends.get?user\_id={friend['id']}&fields=status,bdate,country,city&count={count}&order=hints&access\_token={token}&v=5.81"

friends\_friend = fetch\_request(friends\_url, "items")

for friend\_f in friends\_friend:

user\_f = build\_friend(friend\_f)

friends\_data.append(user\_f)

print(user\_f)

return friends\_data

def clustering\_k\_means(friends\_data, sign, count\_clusters):

friends\_data = list(map(lambda friend: [friend["id"], friend[sign]], friends\_data))

clustering\_data = np.array(friends\_data)

for n in count\_clusters:

kmeans = KMeans(n\_clusters=n)

kmeans.fit(clustering\_data)

plt.scatter(clustering\_data[:,0], clustering\_data[:,1], c=kmeans.labels\_, cmap="rainbow")

plt.scatter(kmeans.cluster\_centers\_[:,0], kmeans.cluster\_centers\_[:,1], color="grey")

plt.title(f"Кластеризация пользователей методом k-средних\nКоличество кластеров = {n}")

plt.xlabel("ID пользователей")

plt.ylabel("Количество фотографий")

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

print("Построение социального графа...")

social\_graph\_friends(210662709, 25)

print("Сбор данных друзей...")

friends\_data = fetch\_friends\_data(210662709, 10)

clustering\_k\_means(friends\_data, "count\_photos", [3, 5, 7])

**Результат выполнения программы:**

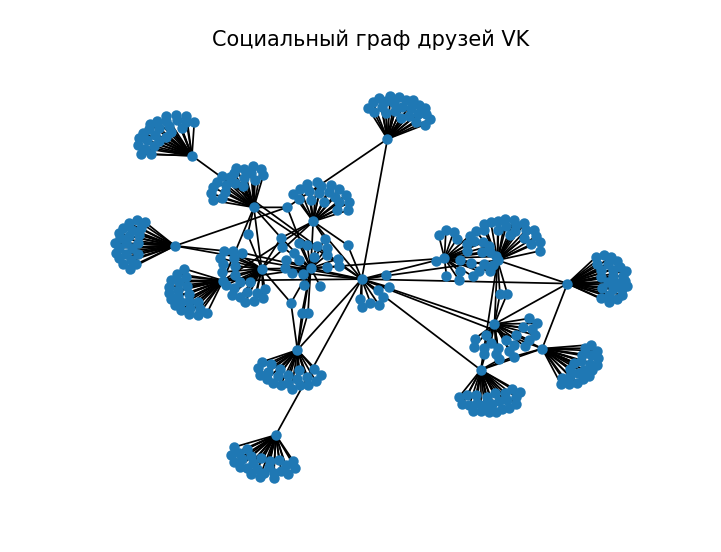
****

Рисунок 1 – Результат работы программы

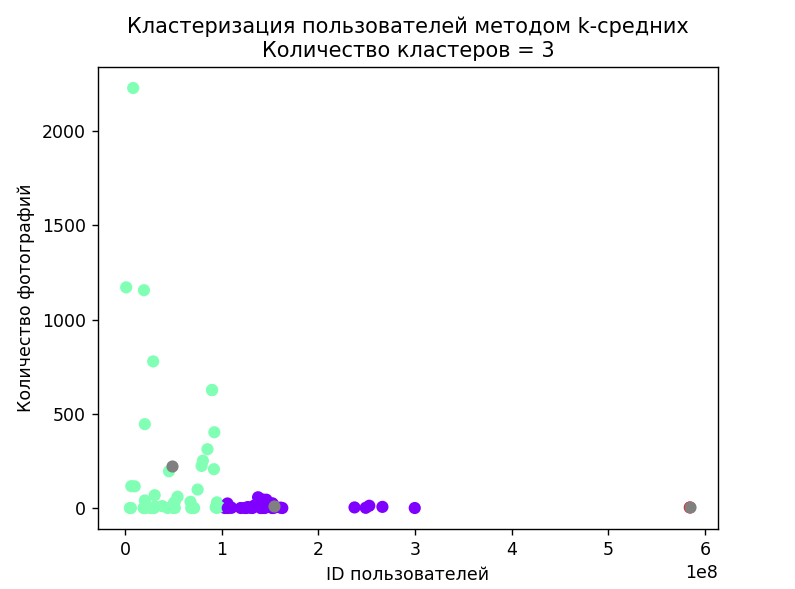


Рисунок 2 - Результат работы программы

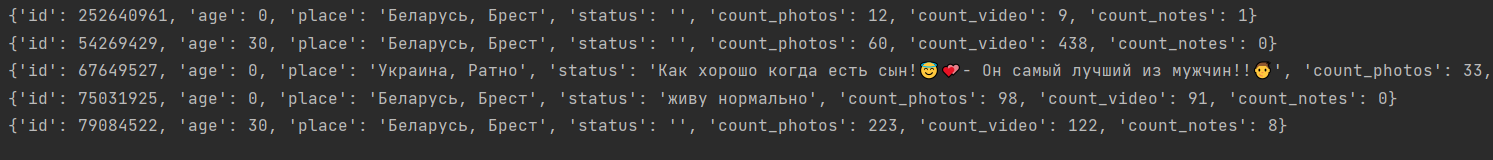


Рисунок 3 - Результат работы программы

**Вывод:** было реализовано задание в соответствии с условиями.