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-
x1xy1cS57U8Zq7uUfWyJdl2IV3iHHO2kzRBp0HXkBEIHdx84ZbM9skNjFBBqGC7kvFVv3xNOiVIbC
cLgC'

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}&ord
er=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={tok
en}&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={toke
n}&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:

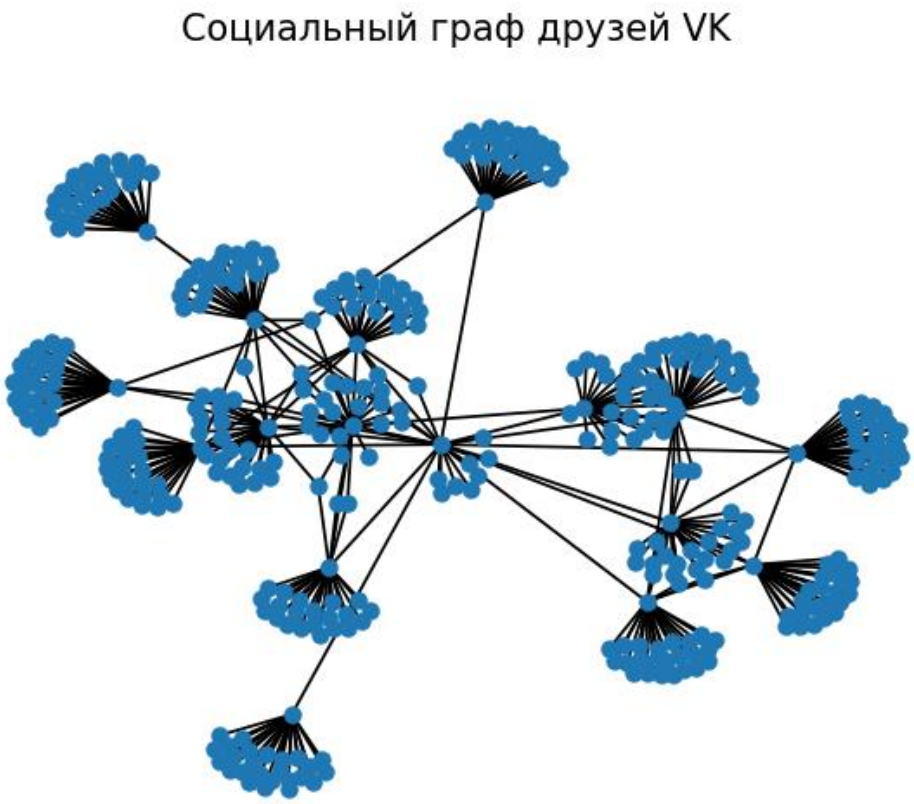

Рисунок 1 – Результат работы программы

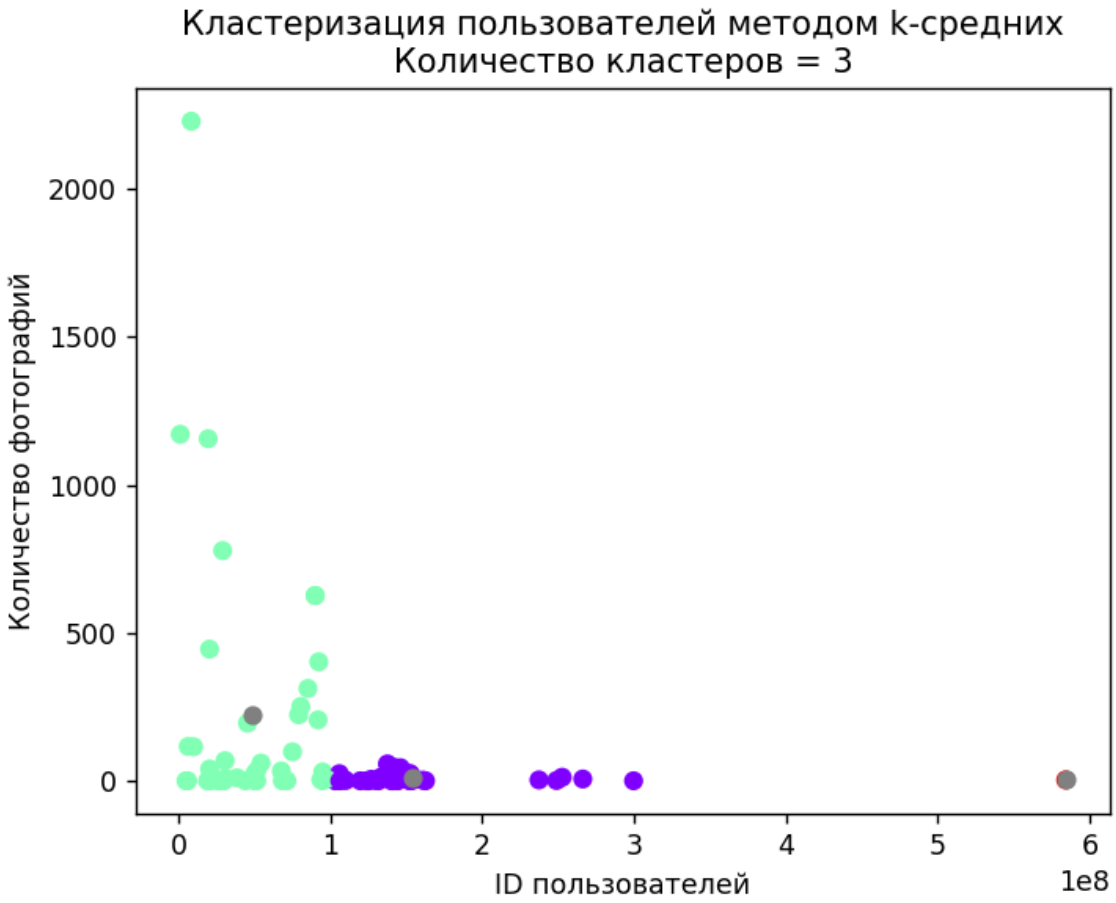

Рисунок 2 - Результат работы программы


```
{'id': 252640961, 'age': 0, 'place': 'Беларусь, Брест', 'status': '', 'count_photos': 12, 'count_video': 9, 'count_notes': 1}
{'id': 54269429, 'age': 30, 'place': 'Беларусь, Брест', 'status': '', 'count_photos': 60, 'count_video': 438, 'count_notes': 0}
{'id': 67649527, 'age': 0, 'place': 'Украина, Ратно', 'status': 'Как хорошо когда есть сын!👨‍❤️‍💋‍👨- Он самый лучший из мужчин!!👨', 'count_photos': 33,
{'id': 75031925, 'age': 0, 'place': 'Беларусь, Брест', 'status': 'живу нормально', 'count_photos': 98, 'count_video': 91, 'count_notes': 0}
{'id': 79084522, 'age': 30, 'place': 'Беларусь, Брест', 'status': '', 'count_photos': 223, 'count_video': 122, 'count_notes': 8}
```


Рисунок 3 - Результат работы программы

Вывод: было реализовано задание в соответствии с условиями.