МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное   
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет   
имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики  
Кафедра технической кибернетики

**Отчет по лабораторной работе № 3**

Дисциплина: «Анализ социальных сетей»

Тема: «Кластеризация изображений»

Выполнил: Овчинников И.В.

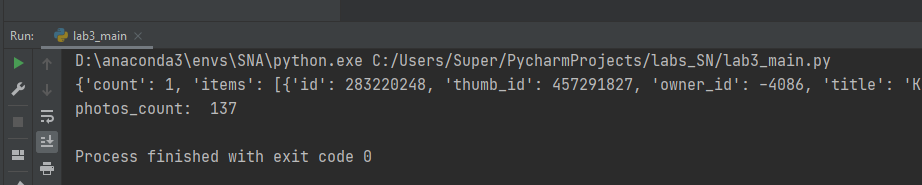
Группа: 6233-010402D

Самара 2021

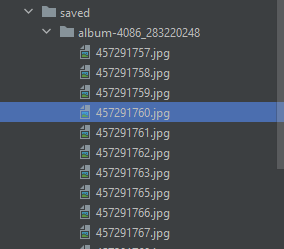
Задание № 1. Скачивание фотографий из социальной сети

Для скачивания фотографий из VK используется vk\_api.

Авторизация в VK, разбор ссылки на альбом с изображениями и получение информации об альбоме:



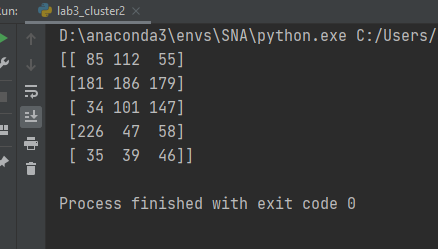
Загрузка изображений из альбома:



Код программы приведен в приложении А.

Задание № 2. Кластеризация цветов на изображении

Получение пяти векторов, где закодированы пять основных цветов в формате RGB, а также построение гистограммы использования этих пяти основных цветов:

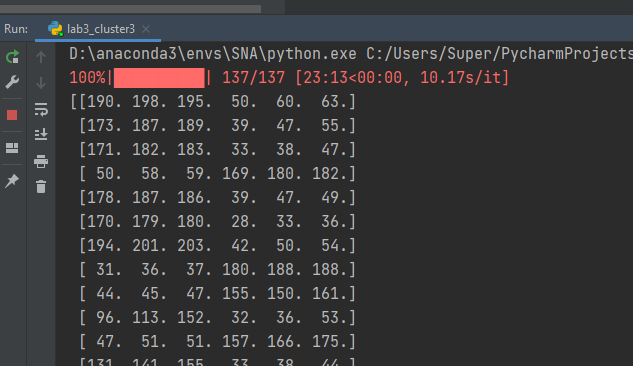


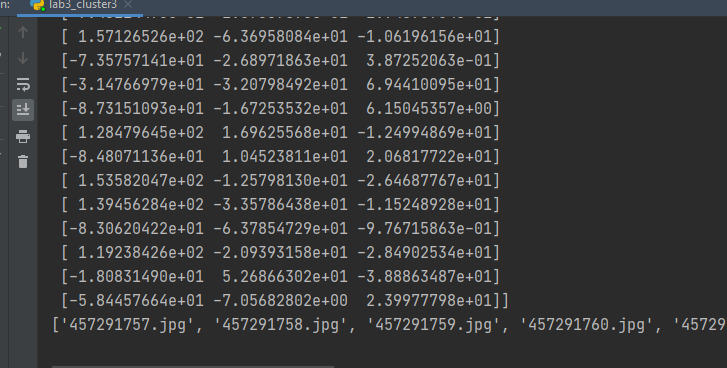


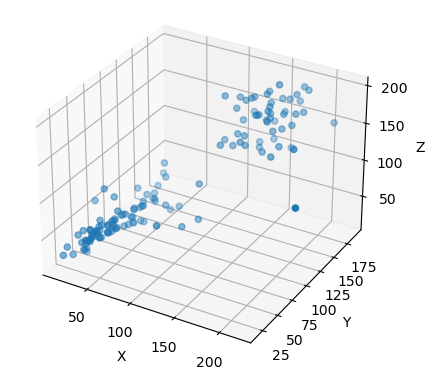
Код программы приведен в приложении Б.

Задание № 3. Кластеризация изображений на основе доминирующих цветов

Обработка всех изображений в директории, понижение размерности вектора характеристик, представление точек изображения на плоскости, где координаты – элементы вектора характеристик:







Код программы приведен в приложении В.

**Приложение А**

**Код программы для задания №1**

import vk\_api  
import time  
import os  
import math  
import requests  
  
  
def auth\_handler():  
 *""" При двухфакторной аутентификации вызывается эта функция.  
 """* # Код двухфакторной аутентификации  
 key = input("Enter authentication code: ")  
 # Если: True ‐ сохранить, False ‐ не сохранять.  
 remember\_device = True  
 return key, remember\_device  
  
  
login, password = 'логин', 'пароль'  
vk\_session = vk\_api.VkApi(  
 login, password,  
 auth\_handler=auth\_handler  
)  
try:  
 vk\_session.auth()  
except vk\_api.AuthError as error\_msg:  
 print(error\_msg)  
  
tools = vk\_api.VkTools(vk\_session)  
  
url = "https://vk.com/album-4086\_283220248"  
# Разбираем ссылку  
album\_id = url.split('/')[-1].split('\_')[1]  
owner\_id = url.split('/')[-1].split('\_')[0].replace('album', '')  
  
response = tools.get\_all("photos.getAlbums", 100, {'owner\_id': owner\_id, 'album\_ids': album\_id})  
photos\_count = response['items'][0]['size']  
print(response)  
print("photos\_count: ", photos\_count)  
  
counter = 0 # текущий счетчик  
prog = 0 # процент загруженных  
breaked = 0 # не загружено из-за ошибки  
time\_now = time.time() # время старта  
  
if not os.path.exists('saved'):  
 os.mkdir('saved')  
photo\_folder = 'saved/album{0}\_{1}'.format(owner\_id, album\_id)  
if not os.path.exists(photo\_folder):  
 os.mkdir(photo\_folder)  
for j in range(math.ceil(photos\_count / 1000)):  
 photos = tools.get\_all("photos.get", 100, {'owner\_id': owner\_id, 'album\_id': album\_id, "count": 1000, "offset": j\*1000, "v": 5.95})['items']  
 # print(photos)  
 for photo in photos:  
 print(photo["id"])  
 counter += 1  
 sizes = photo['sizes']  
 s = photo['sizes'][0]  
 value\_x = 0  
 value\_y = 0  
 for size in sizes: #выбираем самый большой размер  
 if value\_x < size['width']:  
 value\_x = size['width']  
 s = size  
 if value\_y < size['height']:  
 value\_y = size['height']  
 s = size  
 print(s['url'])  
 url\_ = s['url']  
 print('Загружаю фото № {} из {}. Прогресс: {} %'.format(counter, photos\_count, prog))  
 prog = round(100/photos\_count\*counter,2)  
 try:  
 r = requests.get(url\_)  
 with open(f"{photo\_folder}/{photo['id']}.jpg", 'wb') as outfile:  
 outfile.write(r.content)  
 except Exception:  
 print('Произошла ошибка, файл пропущен.')  
 breaked += 1  
 continue  
time\_for\_dw = time.time() - time\_now  
print("\nВ очереди было {} файлов. Из них удачно загружено {} файлов, {} не удалось загрузить.")

**Приложение Б**

**Код программы для задания №2**

import cv2  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
import numpy as np  
  
  
class DominantColors:  
 CLUSTERS = None  
 IMAGE = None  
 COLORS = None  
 LABELS = None  
  
 def \_\_init\_\_(self, image, clusters=3):  
 self.CLUSTERS = clusters  
 self.IMAGE = image  
  
 def dominantColors(self):  
 # read image  
 img = cv2.imread(self.IMAGE)  
 # convert to rgb from bgr  
 img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 # reshaping to a list of pixels  
 img = img.reshape((img.shape[0] \* img.shape[1], 3))  
 # save image after operations  
 self.IMAGE = img  
  
 # using k-means to cluster pixels  
 kmeans = KMeans(n\_clusters=self.CLUSTERS)  
 kmeans.fit(img)  
  
 # the cluster centers are our dominant colors.  
 self.COLORS = kmeans.cluster\_centers\_  
 # save labels  
 self.LABELS = kmeans.labels\_  
 # returning after converting to integer from float  
 return self.COLORS.astype(int)  
  
 def plotHistogram(self):  
 # labels form 0 to no. of clusters  
 numLabels = np.arange(0, self.CLUSTERS + 1)  
  
 # create frequency count tables  
 (hist, \_) = np.histogram(self.LABELS, bins=numLabels)  
 hist = hist.astype("float")  
 hist /= hist.sum()  
  
 # appending frequencies to cluster centers  
 colors = self.COLORS  
  
 # descending order sorting as per frequency count  
 colors = colors[(-hist).argsort()]  
 hist = hist[(-hist).argsort()]  
  
 # creating empty chart  
 chart = np.zeros((50, 500, 3), np.uint8)  
 start = 0  
  
 # creating color rectangles  
 for i in range(self.CLUSTERS):  
 end = start + hist[i] \* 500  
  
 # getting rgb values  
 r = colors[i][0]  
 g = colors[i][1]  
 b = colors[i][2]  
  
 # using cv2.rectangle to plot colors  
 cv2.rectangle(chart, (int(start), 0),(int(end), 50), (r, g, b), -1)  
 start = end  
  
 # display chart  
 plt.figure()  
 plt.axis("off")  
 plt.imshow(chart)  
 plt.show()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 img = 'saved/album-4086\_283220248/457291795.jpg'  
 clusters = 5  
 dc = DominantColors(img, clusters)  
 colors = dc.dominantColors()  
 dc.plotHistogram()  
 print(colors)

**Приложение В**

**Код программы для задания №3**

import os  
from lab3\_task2 import \*  
from tqdm import tqdm  
  
directory = 'saved/album-4086\_283220248'  
filenames = []  
data = []  
for filename in tqdm(os.listdir(directory)):  
 data\_iter = []  
 filenames.append(filename)  
 img = str(directory) + '\\' + filename  
 clusters = 2  
 dc = DominantColors(img, clusters)  
 colors = dc.dominantColors()  
 for i in colors:  
 for j in i:  
 data\_iter.append(j)  
 data.append(data\_iter)  
  
np\_data = np.asarray(data, dtype=np.float32)  
print(np\_data)  
from sklearn.decomposition import PCA  
pca = PCA(n\_components = 3)  
XPCAreduced = pca.fit\_transform(np\_data)  
print (XPCAreduced)  
print(filenames)  
  
xs, ys, zs = np\_data[:, 0], np\_data[:, 1], np\_data[:,2]  
fig = plt.figure()  
ax = fig.add\_subplot(111, projection='3d')  
ax.scatter(xs, ys, zs)  
ax.set\_xlabel('X')  
ax.set\_ylabel('Y')  
ax.set\_zlabel('Z')  
plt.show()