ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ  
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра АСУ

Отчет

о лабораторной работе №2

по дисциплине «Интеллектуальный анализ данных»

на тему: «Применение библиотеки Scikit-Learn для решения задач кластеризации»

Выполнил:

студент группы ИСТ-19а

Шевченко М. В.

Проверили:

Васяева Т. А.

Шуватова Е. А.

Донецк – 2022

**Цель работы:** изучение алгоритмов кластеризации, изучение возможностей библиотеки Scikit-Learn для решения задач кластеризации.

Порядок выполнения работы

1. Выбрать одну из баз данных (согласовать с преподавателем), список которых представлен на сайте http://archive.ics.uci.edu/ml/datasets.php?format=&task=clu&att=&area=&numAtt=&numIns=&type=&sort=nameUp&view=table. Скачать данные с сайта или можно использовать данные, полученные на практике, результаты НИРС или данные для бакалаврской работы.
2. Исходные данные должны храниться в файле. Программным способом считать эти данные и сформировать обучающую и тестовую выборки.
3. Разработать программу на любом языке программирования с использованием любых существующих библиотек. Рекомендован язык Python с использованием библиотек NumPy, Pandas, Scikit-Learn. Программа должна обеспечивать: реализацию метода кластеризации, который указан в задании; визуализацию результатов; тестирование модели; оценка точности модели; сохранение / считывание модели; вывод модели.
4. Согласно варианту заданий (таблица 2.8) построить несколько моделей. В процессе построения моделей, анализируйте ее качество и записывайте результаты в таблицу, указывая значения гиперпараметров.

**Индивидуальное задание:** Вариант 12, Affinity propagation

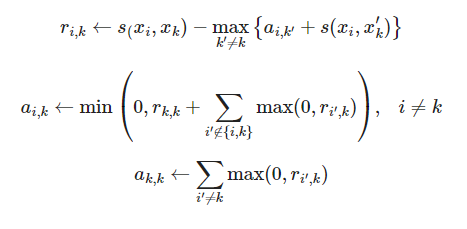
Affinity propagation (метод распространения близости) получает на вход матрицу схожести между элементами датасета *S=N\*N* и возвращает набор меток, присвоенных этим элементам.

В отличие от алгоритма К-средних, данный подход не требует заранее определять число кластеров на которое мы хотим разбить наши данные. Основная идея алгоритма заключается в том, что нам хотелось бы, чтобы наши наблюдения кластеризовались в группы на основе того, как они "общаются" или насколько они похожи друг на друга.

Заведём для этого какую-нибудь метрику "похожести", определяющуюся тем, что S(Xi,Xj)>S(Xi,Xk) если наблюдение Xi больше похоже на наблюдение Xj, чем на Xk. . Простым примером такой похожести будет отрицательный квадрат расстояния S(Xi,Xj) = -||Xi-Xj|^2|

.Теперь опишем сам процесс "общения" для этого заведем две матрицы, инициализируемые нулями, одна из которых Ri,k будет описывать насколько хорошо k-тое наблюдение подходит для того, чтобы быть "примером для подражания" для i-того наблюдения, относительно всех остальных потенциальных "примеров", а вторая — Ai,k будет описывать насколько правильным было бы для -того наблюдения выбрать k-тое в качестве такого "примера".

После этого данные матрицы обновляются по очереди по правилам:



Метод может быть реализован при помощи класса sklearn.cluster.AffinityPropagation библиотеки sklearn. Среди параметров можно выделить:

* damping

Cтепень сохранения текущего значения относительно поступающих значений (взвешенный 1 — демпфирование). Это для того, чтобы избежать числовых колебаний при обновлении этих значений (сообщений)

* max\_iterint

Максимальное количество итераций

**Описание набора данных**

Этот набор данных включает данные для оценки уровня ожирения у людей из стран Мексики, Перу и Колумбии на основе их пищевых привычек и физического состояния. Данные содержат 17 атрибутов и 2111 записей, записи помечены переменной класса NObesity (Уровень ожирения), что позволяет классифицировать данные с использованием значений Недостаточный вес, Нормальный вес, Избыточный вес I уровня, Избыточный вес II уровня, Ожирение I типа. , Ожирение II типа и Ожирение III типа. 77% данных были сгенерированы синтетическим путем с помощью инструмента Weka и фильтра SMOTE, 23% данных были собраны непосредственно от пользователей через веб-платформу.

https://archive.ics.uci.edu/ml/datasets/Estimation+of+obesity+levels+based+on+eating+habits+and+physical+condition+

**Листинг программы**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import AffinityPropagation

from sklearn.decomposition import PCA

import pickle

def save\_model(path, model):

with open('%s.pickle' % path, 'wb') as f:

pickle.dump(model, f)

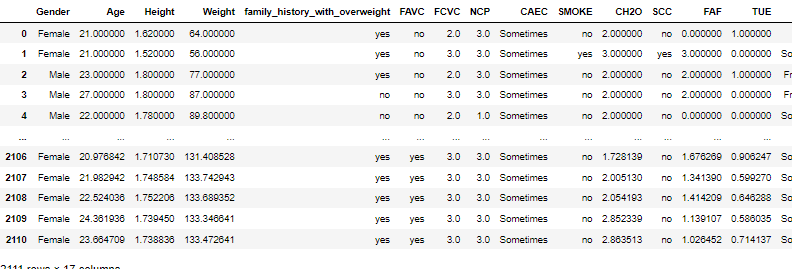
def load\_model(path):

with open('%s.pickle' % path, 'rb') as f:

return pickle.load(f)

df = pd.read\_csv('D:\data2.csv')

df



level = pd.factorize(df['NObeyesdad'])[0]

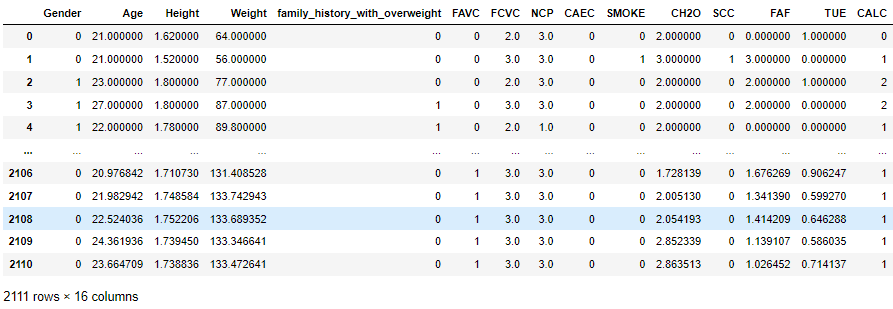
level\_count = pd.factorize(df['NObeyesdad'])[1]

df = df.drop('NObeyesdad', axis=1)

s\_cols = ['Gender', 'family\_history\_with\_overweight','FAVC','CAEC','SMOKE','SCC','CALC','MTRANS']

df[s\_cols] = df[s\_cols].apply(lambda x: pd.factorize(x)[0])

df



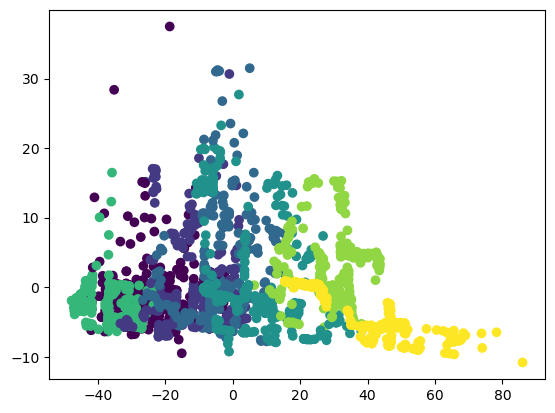
pca = PCA(n\_components=2).fit(df)

df2d = pca.transform(df)

plt.scatter(df2d[:,0], df2d[:,1], c=level)

plt.show()

print('количество классов:', len(level\_count))



количество классов: 7

damping=[0.7, 0.85,0.99]

max\_iter = [100,200,400]

for d in damping:

for mi in max\_iter:

af = AffinityPropagation(damping=d, max\_iter=mi).fit(df)

cluster\_centers\_indices = af.cluster\_centers\_indices\_

labels = af.labels\_

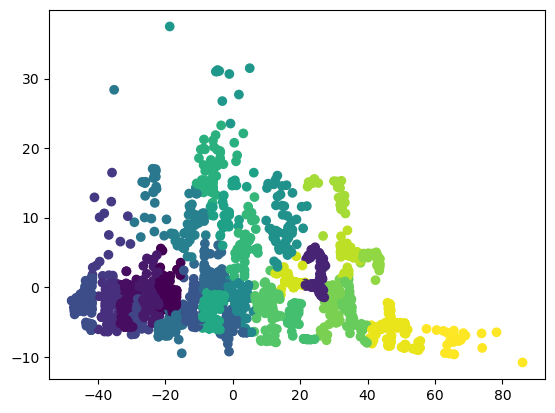
n\_clusters = len(cluster\_centers\_indices)

plt.scatter(df2d[:,0], df2d[:,1], c=labels)

plt.show()

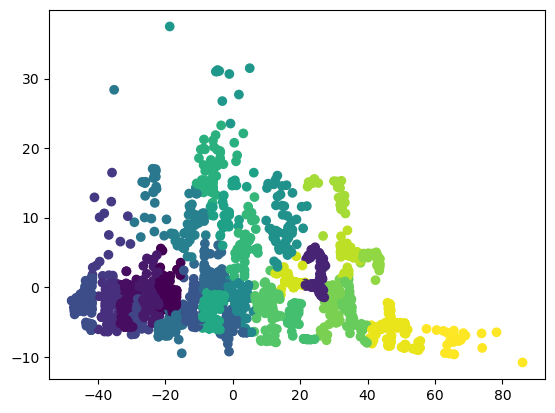
plt.clf()

print('damping =',d,'; max\_iter=',mi,'\nКоличество кластеров:',n\_clusters,'\n\n')



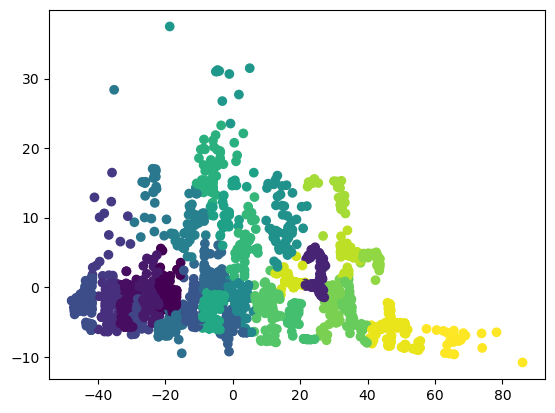
damping = 0.7 ; max\_iter= 100

Количество кластеров: 31



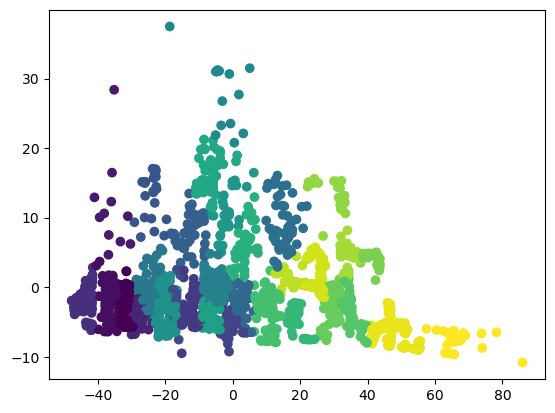
damping = 0.7 ; max\_iter= 200

Количество кластеров: 31



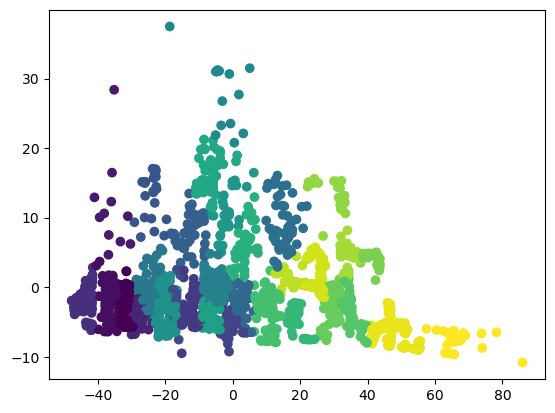
damping = 0.7 ; max\_iter= 400

Количество кластеров: 31



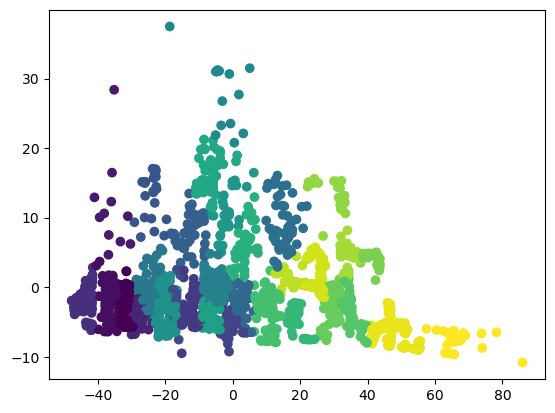
damping = 0.85 ; max\_iter= 100

Количество кластеров: 31



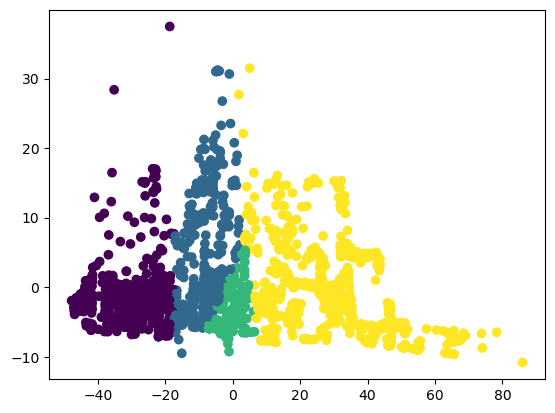
damping = 0.85 ; max\_iter= 200

Количество кластеров: 31



damping = 0.85 ; max\_iter= 400

Количество кластеров: 31



damping = 0.99 ; max\_iter= 400

Количество кластеров: 4