**Задание**

Изображение выглядит как текст

Автоматически созданное описание

**Задание 1**

Используем модуль StandardScaler из библиотеки sklearn, чтобы создать стандартизированный датасет. Сравним, как выглядят данные в обычном и стандартизованном датасетах.

Обычный:

Изображение выглядит как текст, стол

Автоматически созданное описание

Стандартизированный:

Изображение выглядит как текст

Автоматически созданное описание

Сравним 2 модели KMeans. Первая будет использовать не стандартизированные данные, вторая – стандартизированные. Будем сравнивать результаты разбиения для этих 2 моделей в зависимости от максимального числа итераций. Для оценки качества разбиения будем использовать артибут класса KMeans под названием “inertia\_”, который представляет собой сумму квадратов расстояний сэмплов до их ближайших центров кластеров.

Представим результаты опытов в виде таблицы. Первый столбец – максимальное число операций, второй – inertia для не стандартизированных данных, третий – inertia для стандартизированных данных. Изображение выглядит как текст

Автоматически созданное описание

**Вывод**

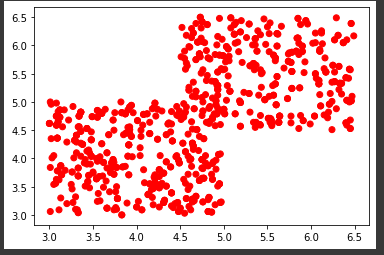
Заметим, что начиная с максимального числа итераций, равного 4, для наших моделей не изменяется метрика inetria, что говорит о быстрой сходимости. При этом сходимость для стандартизированного датасета достигается раньше, начиная с 2 максимальной итераций, что говорит о том, что использование стандартизированного датасета является более предпочтительным.

Также, заметим, что значение метрики inetria для стандартизированного датасета сильно меньше, чем для не стандартизированного, что также говорит о том, что предпочтительнее использовать стандартизированные данные.

**Задание 2**

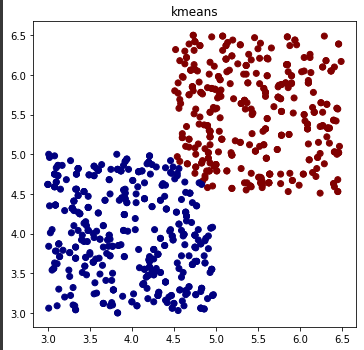
Возьмем датасет ckustering\_1.csv

Изанчально данные выглядят так:

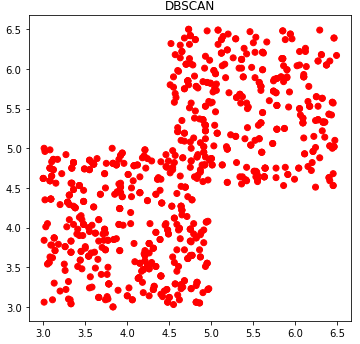


Можно предположить, что оптимальнее всего будет разделить данный датасет на 2 кластера, представляющих собой 2 прямоугольника.

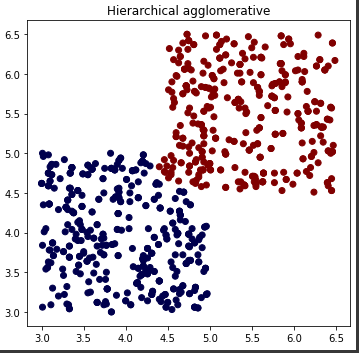
С помощью k-means для него было получено следующее разбиение:



С помощью DBSCAN для него было получено следующее разбиение:



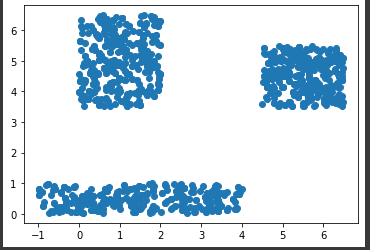
С помощью иерархической кластеризации было получено разбиение:



Как мы видим, k-means и иерархическая кластеризация справились со своей задачей хорошо, в то время как DBSCAN не смог выделить 2 кластера из-за того, что границы кластеров пересекались.

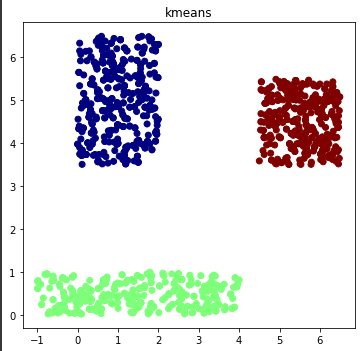
Перейдем к clustering\_2.csv

Данные выглядят следующим образом:

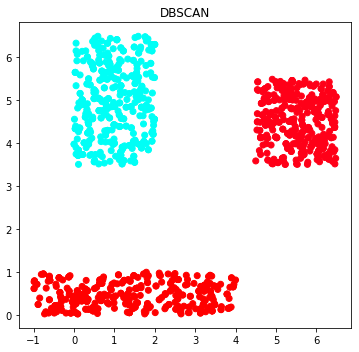


Можно предположить, что оптимальнее всего будет разбить данный датасет на 3 кластера.

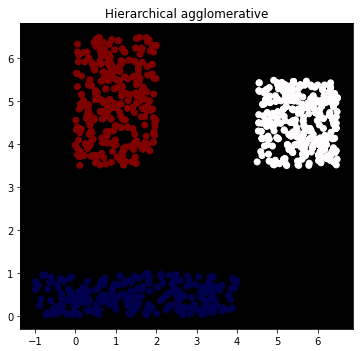
С помощью k-means было получено следующее разбиение:



С помощью DBSCAN было получено следующее разбиение:



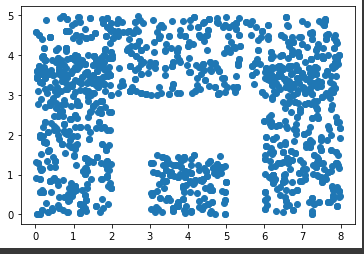
С помощью иерархической кластеризации было получено следующее разбиение: (белые точки были не видны на белом фоне, поэтому фон был изменен на черный)



Тут все методы смогли выделить 3 кластера.

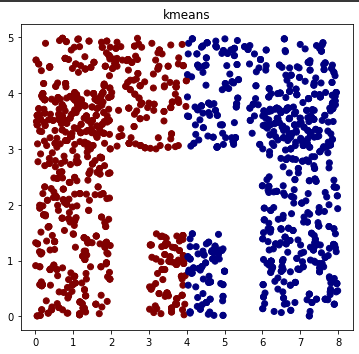
Перейдем clustering\_3.csv

Данные выглядят следующим образом:

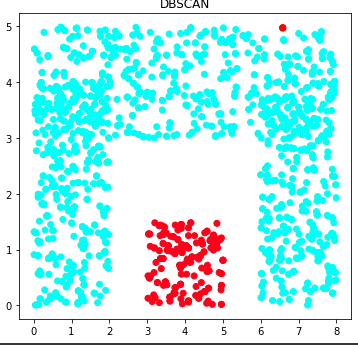


Мы можем предположить, что оптимальнее будет разбить на 2 кластера, где один из них с 3 сторон окружен другим.

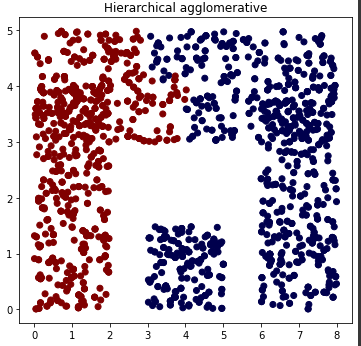
С помощью k-means для него было получено следующее разбиение:



С помощью DBSCAN:



С помощью иерархической кластеризации было получено следующее разбиение:



Заметим, ни один из методов не справился с данной задачей идеально. K-means и иерархическая кластеризация смогли выделить 2 кластера, но кластеры оказались не такими, какими мы ожидали. Метод DBSCAN выделил кластеры почти так, как мы хотели, но засунул лишний элемент в другой кластер.

**Вывод**

На датасетах clustering\_1 и clustering\_2 методы k-means и иерархическая кластеризации показали себя одинаково хорошо. В датасете clustering\_3 оба эти метода выделили 2 кластера, но не такие, какие мы предполагали. Метод DBSCAN показал себя хорошо на датасетах clustering\_2 и clustering\_3, но на датасете clustering\_1 не смог выделить 2 кластера из-за пересечения.

У нас не получилось выделить лучший метод. Целесообразность использования того или иного метода зависит от начального датасета, который мы хотим разбить на кластеры.

**Задание 3**

Возьмем изображение размером 425 х 288 пикселей (тут использовал фотошоп).



Построим изображение с цветами из сокращенной палитры при выделении всего 5 кластеров для палитры:

Изображение выглядит как текст, человек

Автоматически созданное описание

Теперь увеличим число кластеров до 30:

Изображение выглядит как внешний, небо, человек, автомобиль

Автоматически созданное описание

Теперь возьмем 100 кластеров:

Изображение выглядит как человек, внешний, автомобиль

Автоматически созданное описание

**Вывод**

Можем заметить, что при использовании палитры уже из 100 цветов, наше изображении по качеству приближается к оригинальному изображению.

**Задание 4**

Изначально данные в votes выглядят так:

Изображение выглядит как текст, электроника, черный, клавиатура

Автоматически созданное описание

При построении дендограммы появляется ошибка:



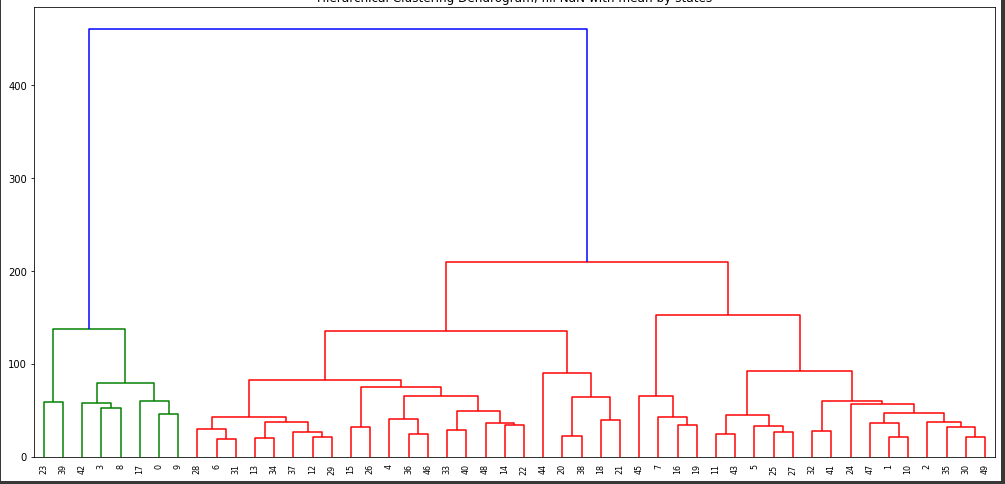
Избавляемся от пустых значений, чтобы избежать ошибки. Заполним их средним для текущего года значением по штатам.

Теперь получили такие данные:

Изображение выглядит как текст, электроника, клавиатура, несколько

Автоматически созданное описание

Можно приступать к построению дендограммы. Используем функцию dendogram из библиотеки sklearn.



**Вывод**

Чем меньше расстояние между двумя штатами, являющимися листами нашей дендограммы, тем меньше разница в голосах за все года. На основе этого происходит разделение штатов на группы. В части групп голоса за республиканцев доминируют, в другой части - нет.