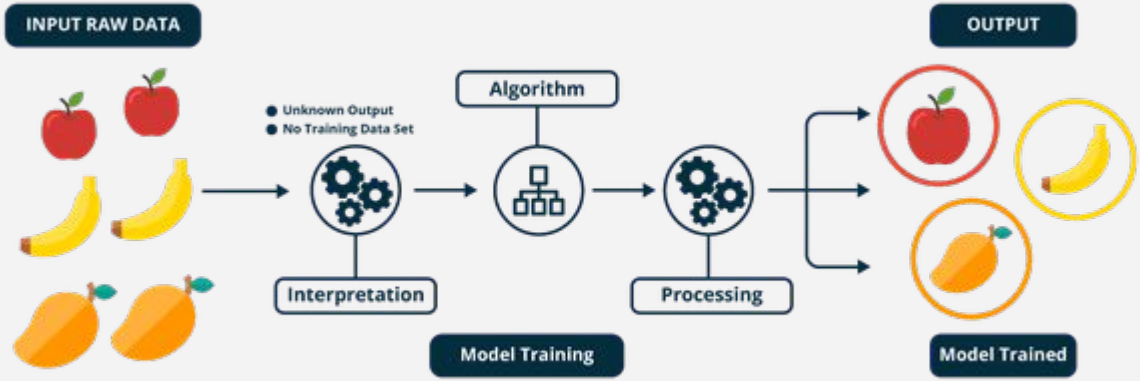
Машинное обучение: несупервизированное обучение

Введение в Unsupervised Learning

Несупервизированное обучение (Unsupervised Learning) – это метод машинного обучения, при котором **алгоритм работает с неразмеченными данными**.



Машинное обучение: несупервизированное обучение

- В отличие от супервизированного обучения, модель **самостоятельно ищет закономерности**, не имея заранее известных меток.

- Основные задачи: **кластеризация, снижение размерности, обнаружение аномалий**.

Ключевые особенности:

- Данные **не размечены**, модель **анализирует их структуру**.
- Нет заранее определённых **классов или целевых переменных**.
- Позволяет **выявлять скрытые зависимости и паттерны** в информации.

Основные задачи несупервизированного обучения

Несупервизированное обучение решает три основные задачи:

1. **Кластеризация**
2. **Снижение размерности (Dimensionality Reduction)**
3. **Обнаружение аномалий (Anomaly Detection)**

Кластеризация

- **Кластеризация (Clustering)** – это метод, который группирует **похожие объекты** в **кластеры**.
- Объекты **внутри кластера** максимально **похожи**, а объекты **из разных кластеров** – **отличаются** друг от друга.
- В отличие от классификации, **заранее классы неизвестны** – алгоритм сам находит их в данных.

Как работает кластеризация?

1. **Система анализирует данные** – объекты рассматриваются в многомерном пространстве.
2. **Поиск схожих объектов** – вычисляется расстояние между точками.
3. **Разделение на группы** – алгоритм распределяет данные **по кластерам**.

Пример

- В **маркетинге** – сегментация клиентов **по интересам** и **поведению**.
- В **медицине** – анализ ДНК, выделение групп пациентов **с похожими заболеваниями**.
- В **анализе изображений** – группировка фотографий **по их содержанию** (например, коты, собаки, машины).

Снижение размерности (Dimensionality Reduction)

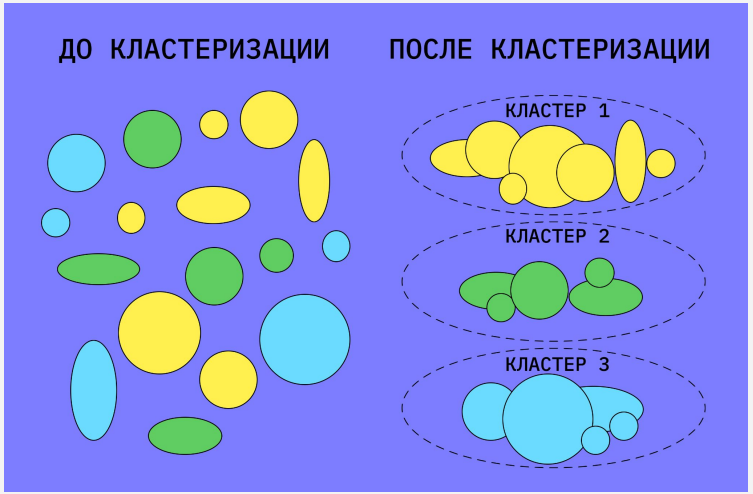
- **Снижение размерности** – это метод, который уменьшает **количество признаков в данных**, сохраняя при этом **ключевую информацию**.

- Позволяет **избежать проблемы "проклятия размерности"**, когда данные становятся слишком сложными для обработки.

- Используется для **визуализации многомерных данных, ускорения вычислений и устранения избыточности**.

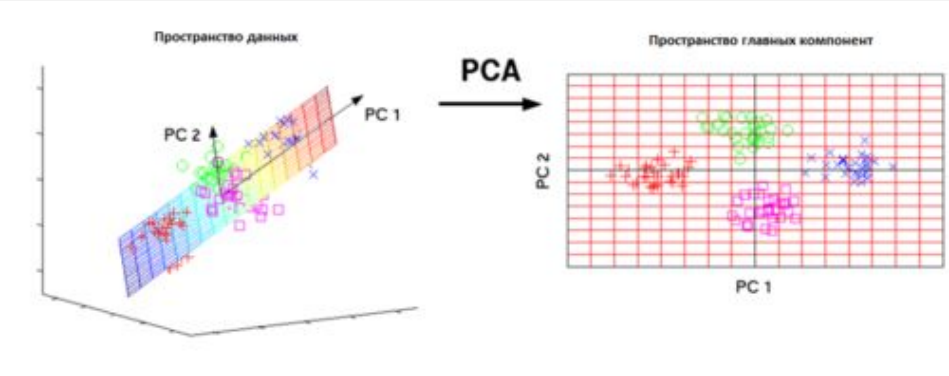
Основные методы

1. **Метод главных компонент** (PCA – Principal Component Analysis)
2. **t-SNE** (t-Distributed Stochastic Neighbor Embedding)
3. **UMAP** (Uniform Manifold Approximation and Projection)



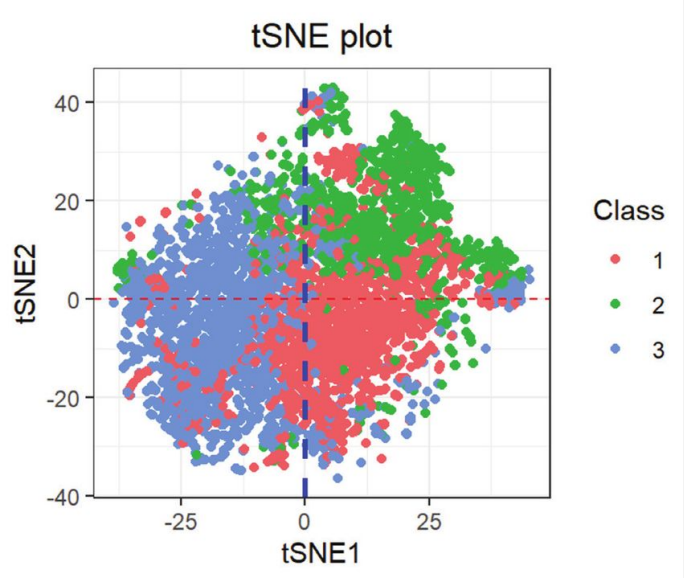
Метод главных компонент (PCA – Principal Component Analysis)

- Находит **главные оси** в данных и проецирует точки на новое пространство.
- Позволяет **сжать** многомерные данные, сохраняя **максимальную дисперсию**.



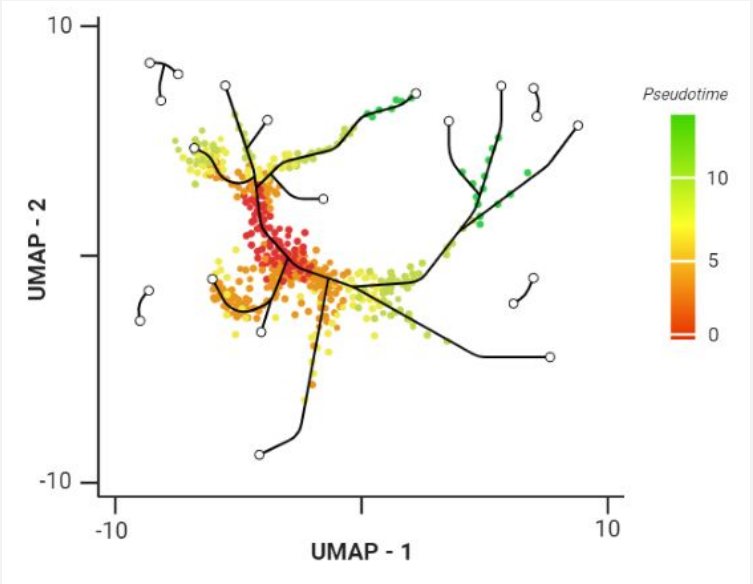
t-SNE (t-Distributed Stochastic Neighbor Embedding)

- Хорошо работает для **визуализации сложных многомерных данных**.
- Применяется для **кластеризации и работы с текстами, изображениями**.



UMAP (Uniform Manifold Approximation and Projection)

- Более **быстрая и точная альтернатива t-SNE**.
- Используется в задачах **биоинформатики, обработки текстов и изображений**.

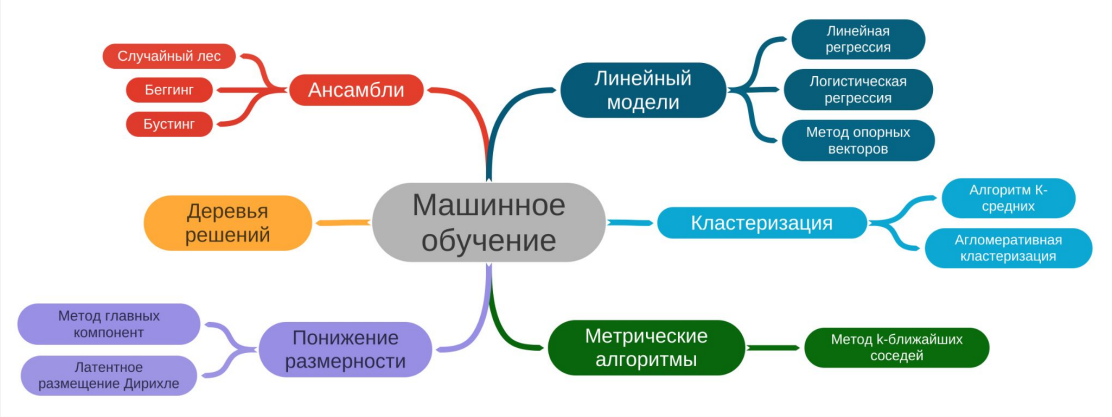


Обнаружение аномалий (Anomaly Detection)

- **Обнаружение аномалий** – это метод, который находит **необычные или редкие объекты** в данных.
- Используется для **выявления мошенничества, диагностики неисправностей и поиска отклонений**.
- Аномалии – это **точки, которые не вписываются в общую структуру данных**.

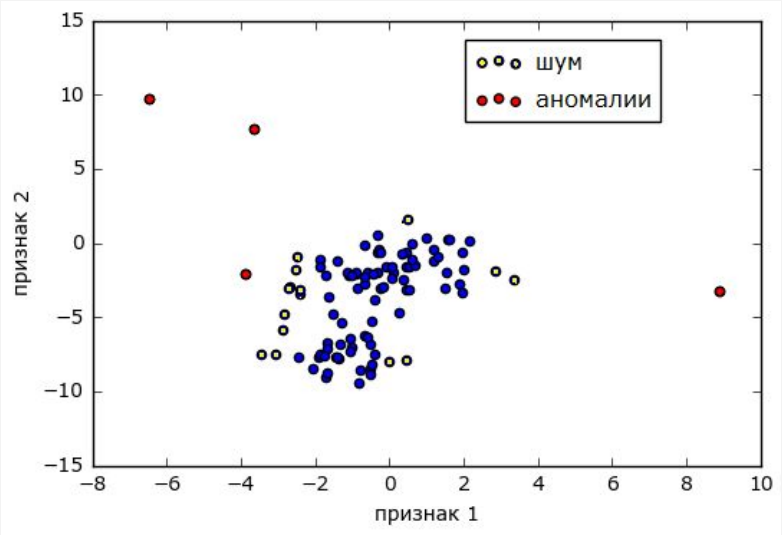
Основные методы обнаружения аномалий

1. Статистические методы
2. Методы машинного обучения
3. Глубокое обучение



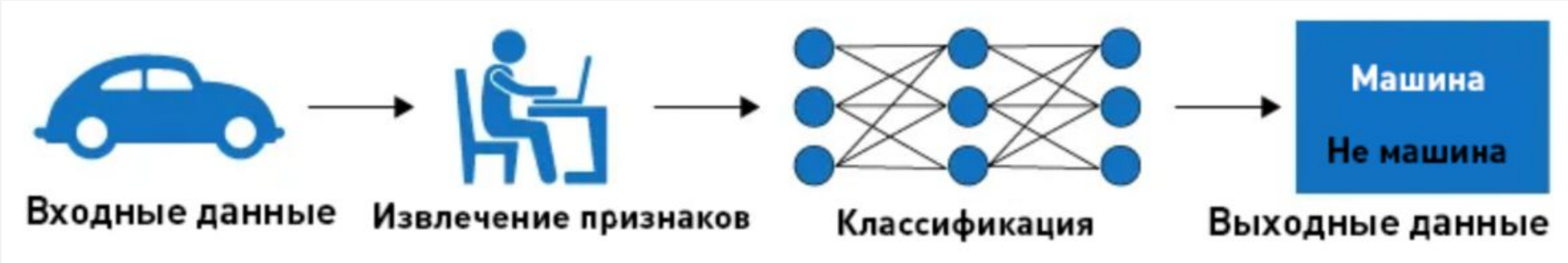
Статистические методы

- Оценка выбросов с помощью **среднего и стандартного отклонения**.
- Вычисление **Z-оценки** и **межквартильного размаха (IQR)**.

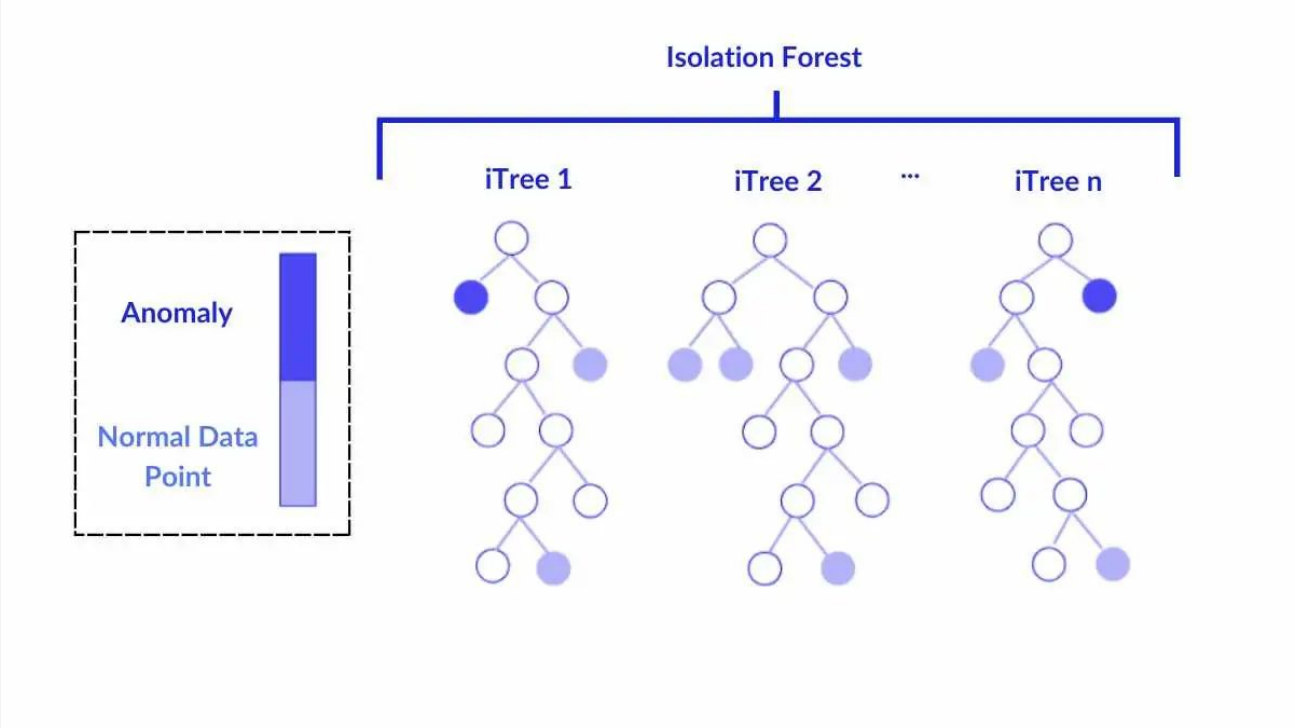


Методы машинного обучения

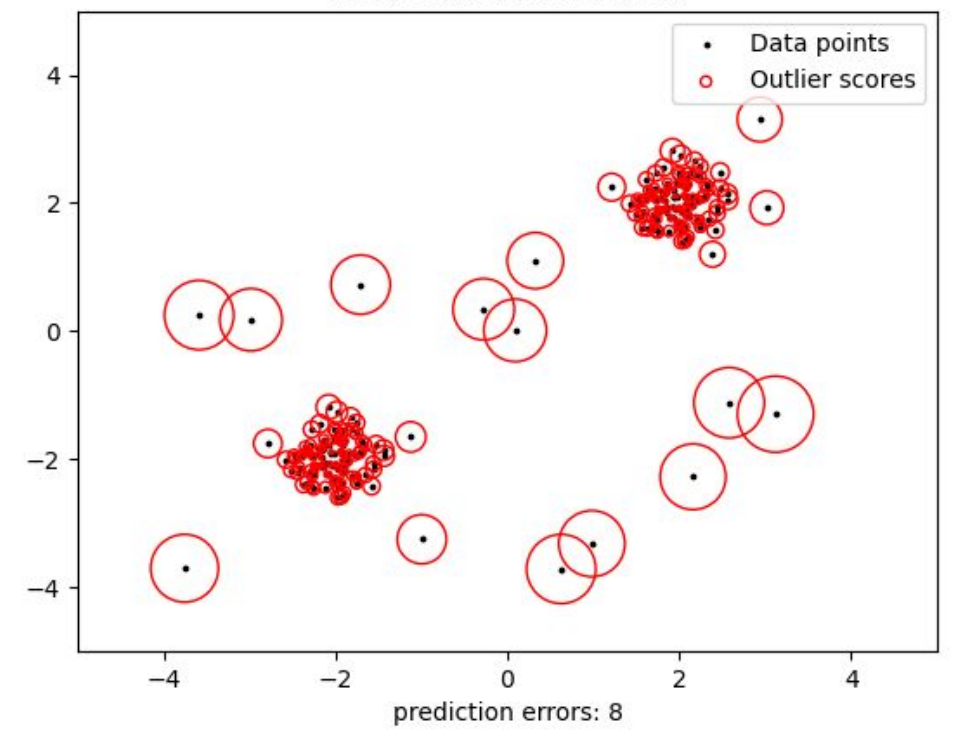
- **Isolation Forest** – строит случайные деревья и оценивает, насколько "легко" изолировать точку.
- **Local Outlier Factor (LOF)** – анализирует плотность точек и сравнивает их с соседями.



Isolation Forest



Local Outlier Factor (LOF)



Глубокое обучение

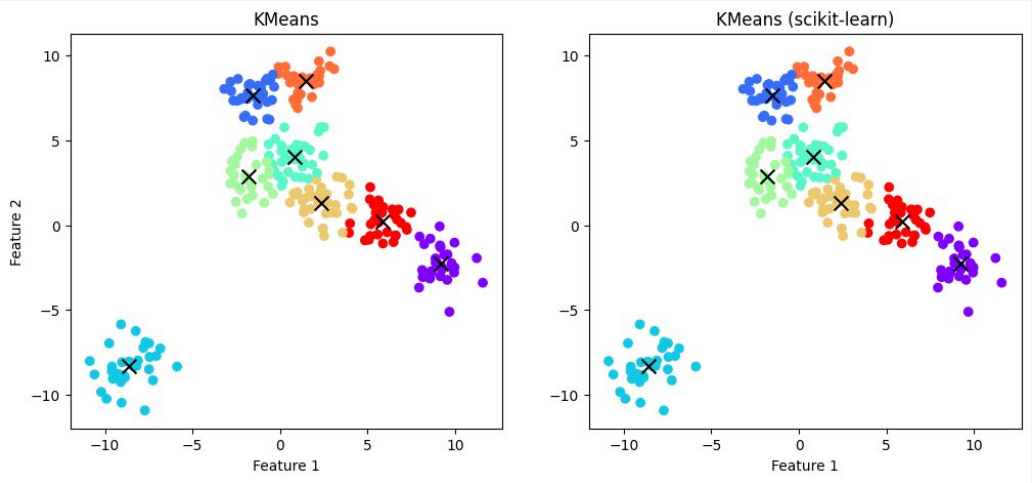
- Автокодировщики (Autoencoders) используются для **поиска аномалий в изображениях и временных рядах**.
- LSTM (Long Short-Term Memory) помогает **выявлять аномалии в последовательностях данных**.

Основные алгоритмы кластеризации

- Существует **несколько алгоритмов кластеризации**, каждый из которых имеет **свои особенности**.
- Выбор метода зависит от **структуры данных, количества кластеров и требуемой точности**.

Популярные алгоритмы кластеризации

1. **K-Means** (К-средних)
2. **DBSCAN** (Density-Based Spatial Clustering of Applications with Noise)
3. **Иерархическая**



Алгоритм K-Means

- **K-Means (Метод K-средних)** – это один из самых популярных алгоритмов кластеризации.
- Разделяет данные на **K кластеров**, где каждая точка **принадлежит кластеру с ближайшим центром**.
- Основан на **поиске центров кластеров (центроидов)** и минимизации расстояния точек до этих центров.

Как работает алгоритм?

1. **Выбираем количество кластеров K** – задаётся вручную.
2. **Инициализируем случайные центры кластеров**.
3. **Назначаем каждую точку** к ближайшему центру.
4. **Пересчитываем центры кластеров** – находим среднее положение точек в каждом кластере.
5. **Повторяем процесс** до тех пор, пока центры кластеров **не перестанут изменяться**.

Код в Python

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

X = [[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]]

kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
labels = kmeans.labels_

plt.scatter([x[0] for x in X], [x[1] for x in X],
c=labels)
plt.show()
```

Особенности K-Means

✅ Хорошо работает на **простых данных с чёткими границами кластеров**.

✅ **Быстрый алгоритм** – подходит для больших данных.

❌ Требует **заранее знать K** (можно подбирать методом "локтя").

❌ Чувствителен к **выбросам** – шум может сильно изменить результат.

❌ Плохо работает, если кластеры имеют **сложную форму или разную плотность**.

Алгоритм DBSCAN – плотностная кластеризация

- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** – это алгоритм кластеризации, который **обнаруживает плотные группы точек** в данных.
- В отличие от K-Means, **не требует заранее задавать количество кластеров** и может **выявлять кластеры сложной формы**.
- Используется в ситуациях, когда данные имеют **разные плотности** и содержат **выбросы (шум)**.

Как работает DBSCAN?

1. **Определяет плотные области данных** – если вокруг точки есть достаточно соседей, она становится **основной точкой кластера**.
2. **Объединяет точки в группы** – если точка **находится рядом** с уже существующим кластером, она к нему присоединяется.
3. **Игнорирует шум** – точки, которые **не принадлежат ни к одному кластеру**, считаются выбросами.

Пример использования DBSCAN

- В **анализе геоданных** – нахождение **городских и сельских районов** по плотности населения.
- В **обнаружении аномалий** – выявление **подозрительных транзакций в банках**.
- В **обработке изображений** – разделение областей на фото **по плотности пикселей**.

Код в Python

```
from sklearn.cluster import DBSCAN
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[1, 2], [2, 2], [2, 3], [8, 7], [8, 8],
[25, 80]])

db = DBSCAN(eps=3, min_samples=2).fit(X)
labels = db.labels_

plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.show()
```

Иерархическая кластеризация (Hierarchical Clustering)

- **Иерархическая кластеризация** – это метод, который **разделяет данные на группы** по принципу **дерева (дендрограммы)**.
- В отличие от K-Means и DBSCAN, **не требует заранее задавать количество кластеров**.
- Хорошо подходит для **маленьких и средних наборов данных**.

Как работает иерархическая кластеризация?

1️⃣ Агломеративный метод (снизу-вверх)

- Начинаем с **каждой точки как отдельного кластера**.
- На каждом шаге **ближайшие кластеры объединяются**.
- Продолжаем, пока не останется **один общий кластер**.

Как работает иерархическая кластеризация?

2️⃣ Дивизивный метод (сверху-вниз)

- Начинаем с **одного большого кластера**.
- Постепенно **разделяем его** на подгруппы.
- Продолжаем, пока **не получим отдельные точки**.

Код в Python

```
from scipy.cluster.hierarchy import dendrogram, linkage
import matplotlib.pyplot as plt

X = [[5, 3], [10, 15], [15, 12], [24, 10], [30, 30],
[85, 70], [71, 80]]

Z = linkage(X, method='ward')

dendrogram(Z)
plt.show()
```

Преимущества и недостатки

✅ Не требует заранее **указывать число кластеров**.

✅ Позволяет **изучить структуру данных** через дендрограмму.

✅ Подходит для **маленьких и средних данных**.

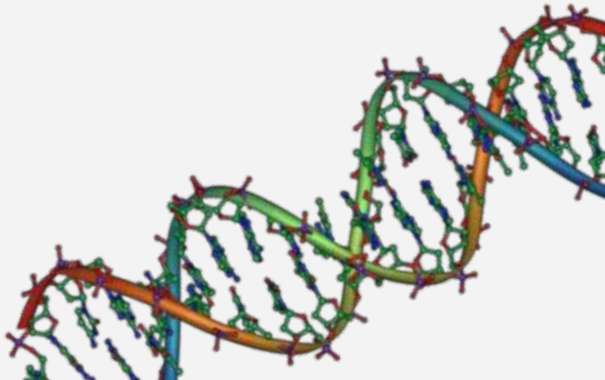
❌ **Медленный** на больших объемах данных.

❌ Может объединять **неправильные кластеры** из-за метода вычисления расстояний.

❌ Не всегда можно определить **оптимальный уровень разбиения**.

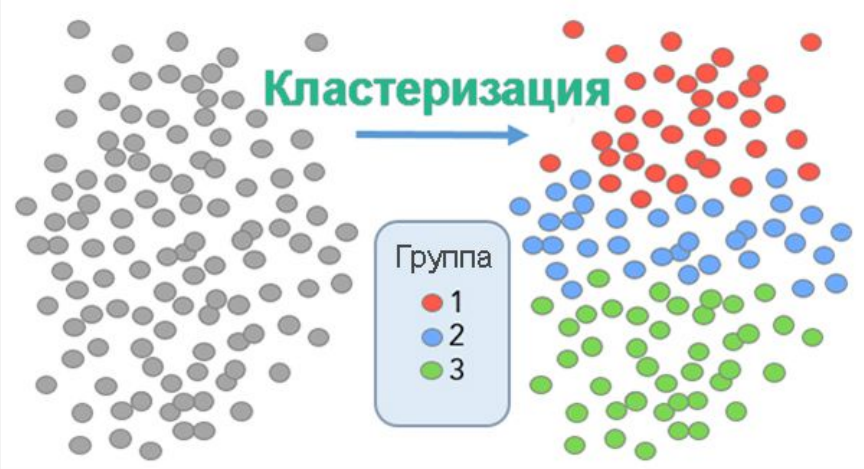
Примеры применения

- В **генетике** – для классификации ДНК-последовательностей.
- В **анализе текста** – группировка новостей по темам.
- В **биоинформатике** – разделение белков на классы.



Ограничения кластеризации

- ***Кластеризация*** – мощный инструмент анализа данных, но у неё есть **ряд ограничений**, которые могут повлиять на качество и интерпретацию результатов.



Основные проблемы кластеризации

1️⃣ Выбор количества кластеров

- Многие алгоритмы (например, K-Means) **требуют заранее задавать число кластеров (K)**.
- Если выбрать **слишком много или слишком мало кластеров**, результаты могут **исказиться**.
- Решение: **Метод локтя (Elbow Method)**, **силуэтный анализ**.

Основные проблемы кластеризации

2️⃣ Чувствительность к выбросам

- **Выбросы (аномальные точки)** могут сильно **испортить результат кластеризации**.
- K-Means, например, **очень чувствителен** к выбросам, так как использует средние значения.
- Решение: **Использовать DBSCAN**, который **лучше обрабатывает шум**.

Основные проблемы кластеризации

3️⃣ Форма кластеров

- Некоторые алгоритмы (например, K-Means) **плохо работают на сложных формах кластеров**.
- Если данные имеют **неоднородную плотность или перекрывающиеся группы**, могут возникнуть ошибки.
- Решение: **Использовать DBSCAN или иерархическую кластеризацию**.

Основные проблемы кластеризации

4️⃣ Высокая размерность данных

- В многомерных данных сложно определять **границы между кластерами**.
- Чем больше признаков, тем **сложнее вычисления** и тем **хуже качество кластеризации**.
- Решение: **Использовать снижение размерности (PCA, t-SNE, UMAP)**.

Основные проблемы кластеризации

5️⃣ Зависимость от метрики расстояния

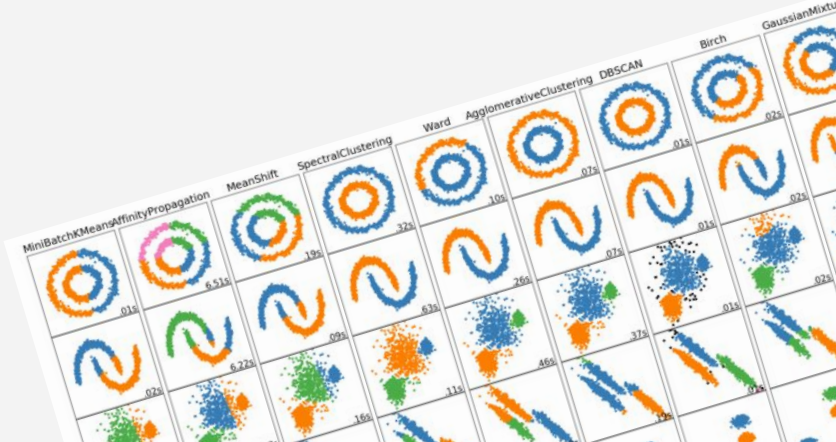
- Алгоритмы часто используют **евклидово расстояние**, которое не всегда подходит.
- В некоторых случаях **манхэттенское расстояние или косинусное сходство** могут давать **лучшие результаты**.

Примеры проблем кластеризации

- В **банковском анализе** – неправильное количество сегментов клиентов может **привести к ошибкам в рекламе**.
- В **биоинформатике** – кластеризация ДНК может **неправильно разделить гены**, если алгоритм выбран неправильно.
- В **анализе текстов** – слова с похожими значениями могут **попасть в разные кластеры**, если метрика расстояния выбрана некорректно.

Выбор алгоритма кластеризации в зависимости от данных

- Разные методы кластеризации **подходят для разных типов данных**.
- Выбор алгоритма зависит от **структуры данных, количества кластеров и их плотности**.



Как выбрать правильный алгоритм кластеризации?

Характеристика данных	**Рекомендуемый алгоритм**
Кластеры имеют чёткую, сферическую форму	K-Means
Данные содержат выбросы и шум	DBSCAN
Кластеры имеют сложную форму	DBSCAN, Иерархическая кластеризация
Нужно изучить структуру данных	Иерархическая кластеризация
Неизвестное число кластеров	DBSCAN, Иерархическая кластеризация
Данные многомерные	PCA + K-Means
Кластеры пересекаются	Gaussian Mixture Model (GMM)

Графическая схема выбора алгоритма

- Если **кластеры сферические → K-Means**.
- Если **есть шум и выбросы → DBSCAN**.
- Если **нужно увидеть иерархию → Иерархическая кластеризация**.
- Если **данные многомерные → PCA + кластеризация**.