

# Лекция 3

# SciKit Learn

- Модуль Python в котором содержится большое количество популярных методов машинного обучения
- Имеет открытый исходный код
- Содержит как сами методы для построения моделей машинного обучения, так и вспомогательные алгоритмы
- Может работать с NumPy и Pandas

# Предобработка данных

# Предобработка данных

- Зачастую, исходные данные непригодны сразу для анализа.
- Стандартные процедуры предобработки данных:
  - Нормировка
  - Стандартизация
  - Устранение отсутствующих значений
  - Преобразование категориальных данных
  - Преобразование транзакционных данных
  - Векторизация текста
  - и много других

# Нормировка [0;1]

- Разные признаки могут иметь разный порядок величин, из-за чего затрудняется построение модели и последующий анализ.
- Например, что можно сказать по этой линейной модели:  $30x + 0.7y = z$
- Самый простой способ, привести все признаки к одному диапазону [0; 1]:

$$\hat{X} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

- В sklearn за нормировку отвечает **MinMaxScaler**

# Нормировка по модулю

- Иногда необходимо сохранять знак величины, и в таких случаях `MinMaxScaler` не подходит.
- Решение, разделить значение каждого признака на максимально значение по модулю и привести признак к диапазон `[-1; 1]`:

$$\hat{X} = \frac{X}{\max(|X|)}$$

- Для этого можно использовать `MaxAbsScaler`

# Устойчивая к выбросам нормировка

- Выбросы в данных могут “испортить” нормировку, и сжать несколько точек в одну.
- Решение, центрировать относительно медианы и разделить на разницу между 1 и 3 квартилью (25 и 75 процентиль):

$$\hat{X} = \frac{X - \tilde{X}}{Q_3 - Q_1}$$

- Для этого можно использовать **RobustScaler**

# Стандартизация

- При нормировке, статистические характеристики признаков различаются. Но некоторые методы и алгоритмы лучше работают, если признаки имеют одинаковое мат. ожидание и стандартное отклонение
- Процедура стандартизации заключается в центрировании по среднему, и деление на стандартное отклонение:

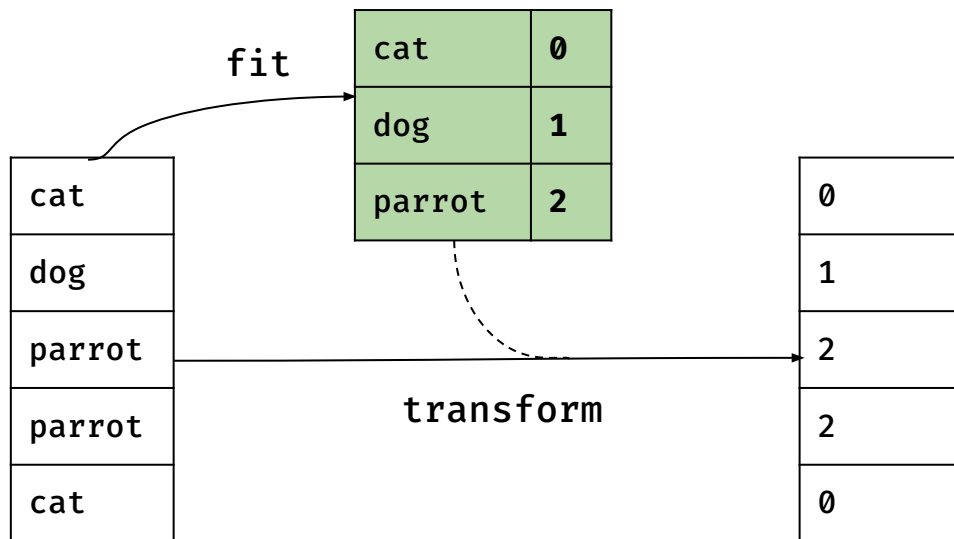
$$\hat{X} = \frac{X - \mathbf{E}X}{\sigma X}$$

- Для этого можно использовать **StandardScaler**



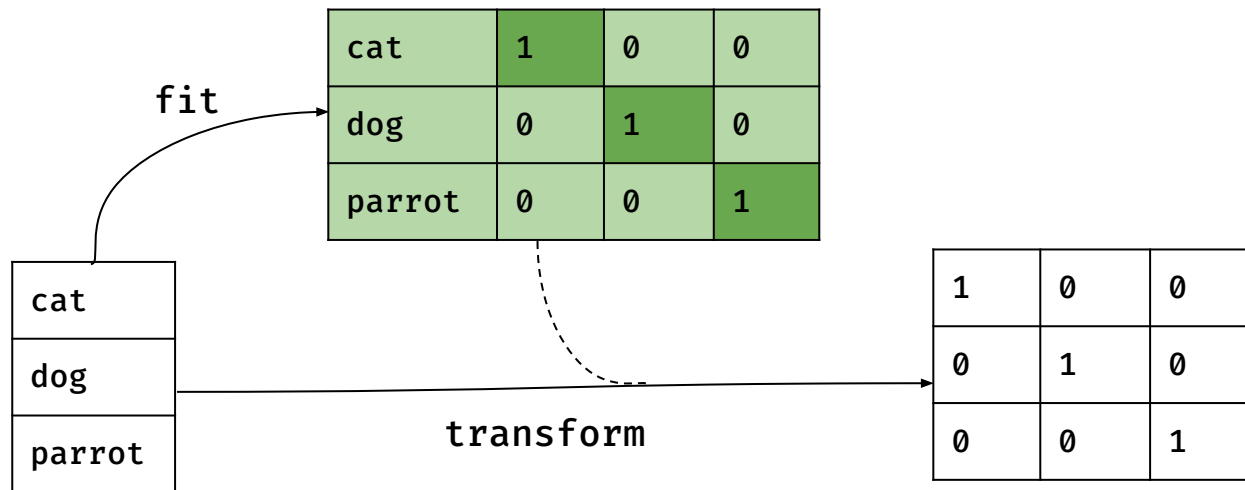
# Кодирование меток классов

- В наборах данных, метки классов часто кодируются в строки. Можно собрать словарь строк, и каждой строке сопоставить порядковый номер.
- В sklearn используется **LabelEncoder**.



# Унитарное кодирование

- **LabelEncoder** плохо подходит в случае, если классов  $>2$ , так как моделям тяжело подстроиться под множество дискретных значений.
- Можно использовать **LabelBinarizer**, который каждую метку класса кодирует индивидуальным унитарным кодом.



# OneHot кодирование

- **OneHotEncoder** позволяет кодировать сразу несколько категориальных меток в виде одного вектора.
- Размер вектора - суммарное количество всех категорий. Для каждой категории сопоставляется свой индекс в векторе.
- В векторе по соответствующему индексу стоит 1, если данная категория есть в наблюдении.

## Еще способы предобработки

- **FunctionTransformer** - позволяет применять собственную функцию преобразования. Например, можно получить модуль каждого признака (`np.abs`).
- **KBinsDiscretizer** - позволяет привести количественный признак к дискретному, разделив диапазон значений на интервалы.
- **OrdinalEncoder** - позволяет преобразовать категориальный признак в порядковый. Например, категории `[-2, 3, 10]` будут преобразованы в `[0, 1, 2]`.
- **PolynomialFeatures** - создает новые признаки за счет комбинаций всех элементов полинома N-й степени. Например, для `[a, b]` и `N = 2` будут получены признаки `[1, a, b, a2, b2, ab]`

# Кодирование текста

1. Сделать словарь где каждому слову сопоставлено число
2. Сделать словарь где каждому слову соответствует индекс в векторе (OneHot)
3. Нейросетевая модель с анализом семантики по типу word2vec

Пример текста «Hello World and Machine Learning!».

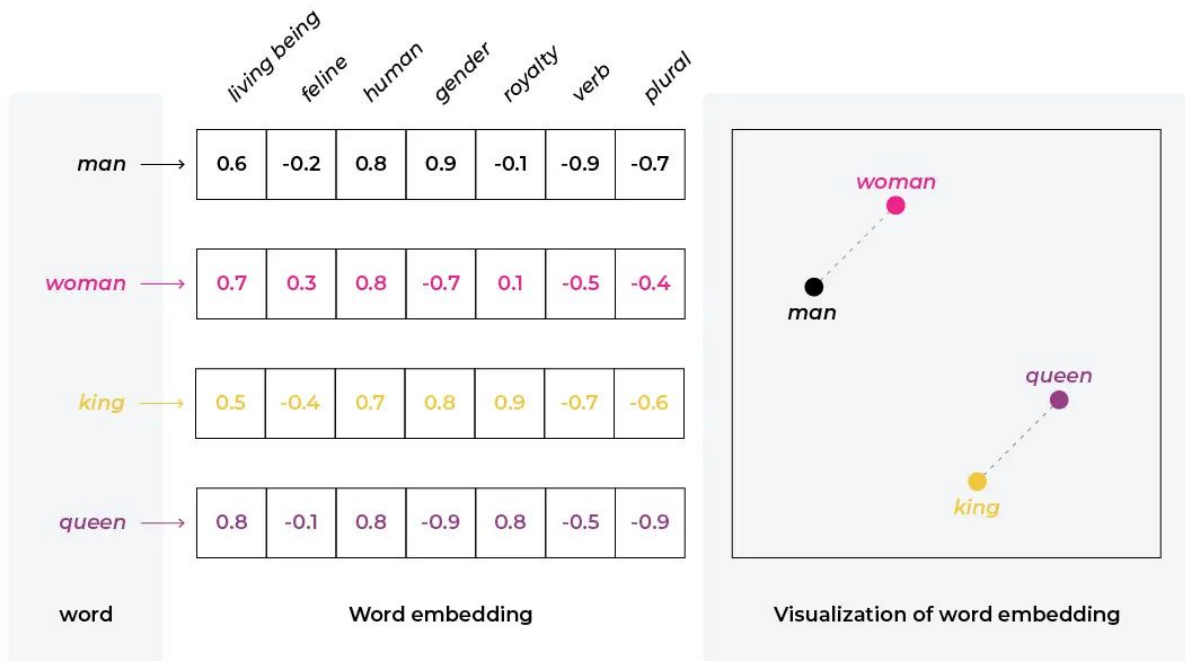
Кодирование 1-м способом -> [2, 7, 1, 5, 4]

Кодирование 2-м способом -> [0, 1, 1, 0, 1, 1, 0, 1]

Слово	Код	Индекс
#	-1	0
And	1	1
Hello	2	2
Goodbye	3	3
Learning	4	4
Machine	5	5
Not	6	6
World	7	7

# word2vec (1)

Кодирует каждое слово вектором определенной размерности таким образом, что семантически близкие слова находятся рядом в кодированном пространстве.



## word2vec (2)

king - man + woman  $\approx$  queen



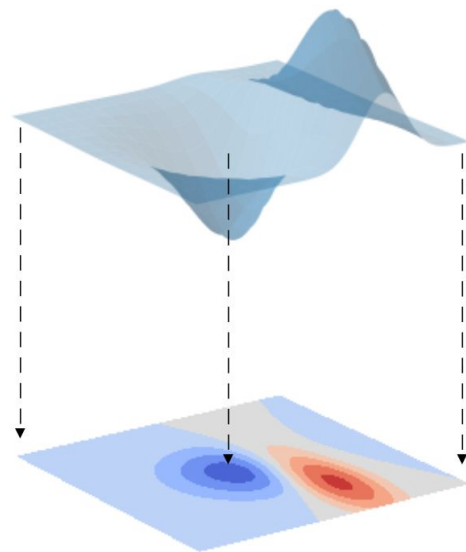
Подробнее: <https://habr.com/ru/articles/446530/>

# Понижение размерности



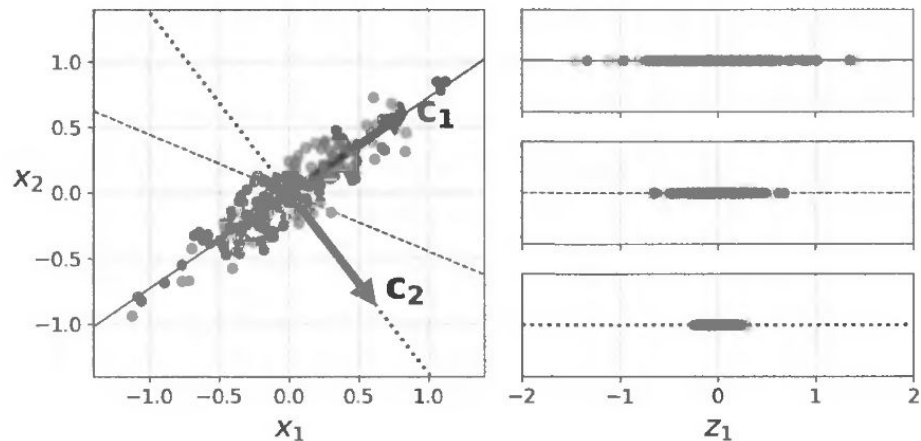
# Понижение размерности

- Понижение размерности - такое преобразование данных, при котором уменьшается количество признаков.
- Часто в наборах данных бывает огромное количество признаков ( $>50$ ), из-за чего обработка и анализ данных усложняются.
- Понижение размерности можно использовать для:
  - Выделение новых значимых признаков
  - Отброс незначимых признаков
  - Устранение шумов
  - Визуализация многомерного пространства



# Метод главных компонент - PCA

- PCA - наиболее популярный метод понижения размерности
- Идея алгоритма заключается в нахождении таких осей, на долю которых приходятся самые большие величины дисперсии в наборе данных. Такие оси называются главными компонентами (principle components)
- Первая компонента описывает наибольшую дисперсию, вторая следующую по убыванию величину дисперсии, и т.д.

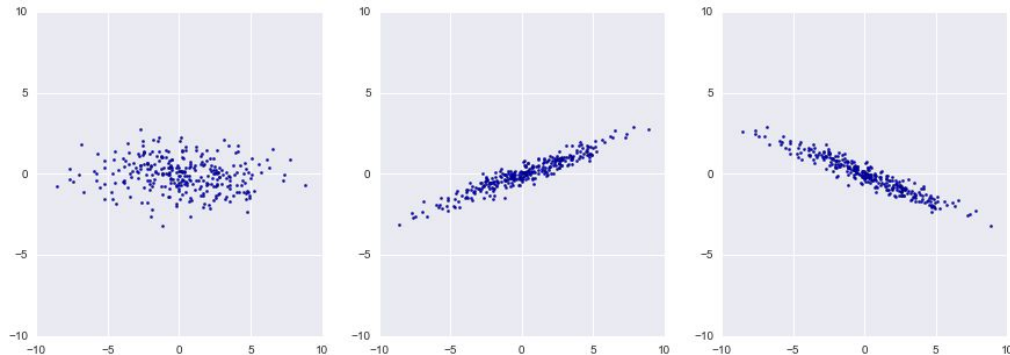


# Ковариационная матрица

- При описании многомерной случайной величины, расположение центра (мат. ожидания) определяется как мат. ожидание проекций, но для описания формы, необходимо использовать ковариационную матрицу  
В общем виде формула  $i, j$  элемента ков. матрицы:

$$\begin{aligned} Cov(X_i, X_j) &= \mathbf{E} [(X_i - \mathbf{E}X_i) \cdot (X_j - \mathbf{E}X_j)] = \\ &= \mathbf{E} (X_i X_j) - \mathbf{E} (X_i) \mathbf{E} (X_j) \end{aligned}$$

- При нулевом мат. ожидании каждого признака:  $Cov(X_i, X_j) = \mathbf{E} (X_i X_j)$



# Нахождение главных компонент

- Так как ковариационная матрица описывает форму набора данных, то ее можно использовать для нахождения компонент.
- Предположим, проекция вектора  $X$  на единичный вектор  $v$ :  $X^* = v^T X$
- Тогда дисперсия вектора при мат. ожидании 0:

$$Var(X) = \mathbf{E}(XX^T)$$

- И можно рассчитать дисперсию проекции как:

$$Var(X^*) = \mathbf{E}(X^*X^{*T}) = \mathbf{E}\left((v^T X)((v^T X)^T)\right) = v^T \mathbf{E}(XX^T) v = v^T Var(X) v$$

- Согласно отношению Рэлея для ков. матриц. Проекция максимальная, если проекция на собс. вектор, а дисперсия соответствует собс. числу

# Матрица преобразования

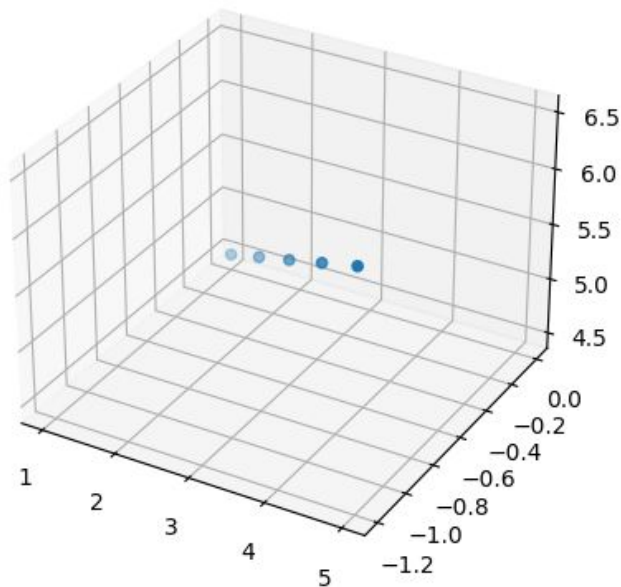
- Так как дисперсия максимальная при проекции на собственные вектора, то собственные вектора ковариационной матрицы являются главными компонентами.
- Предположим, что были получены собственные вектора  $v_1, v_2, v_3 \dots$  и собственные числа  $\lambda_1, \lambda_2, \lambda_3 \dots$ , причем  $\lambda_1 > \lambda_2 > \lambda_3$ . То можно составить матрицу  $W$  состоящую из  $n$  первых собственных векторов-столбцов для проекции набора данных в пространство размерности  $n$ .
- Преобразование данных:  $X^* = XW$
- Обратное преобразование:  $X = X^*W^T$

# Пример PCA (1)

Исходные данные X'

1	0	4.5
2	-0.3	5
3	-0.6	5.5
4	-0.9	6
5	-1.2	6.5

Mean	3	-0.6	5.5
Var	2	0.18	0.5



Стандарт. данные X

-1.41	1.41	-1.41
-0.70	0.70	-0.70
0	0	0
0.70	-0.7	0.70
1.41	-1.41	1.41

$$\frac{6.5 - 5.5}{\sqrt{0.5}} = \frac{1}{0.707}$$

# Пример PCA (2)

Несмещенная ков. мат.

$$CM = X^T * X / 4$$

1.25	-1.25	1.25
-1.25	1.25	-1.25
1.25	-1.25	1.25

Собственные вектора


-0.82	0	0.57
-0.40	0.70	-0.57
0.40	0.70	0.57

Мат. трансформации  
W

0.57	0
-0.57	0.70
0.57	0.70

Собственные числа

-2e-16	2e-16	3.65
--------	-------	------


$$\frac{(-1.414) \cdot 1.414 + (-0.707) \cdot 0.707 + 0 \cdot 0 + 0.707 \cdot (-0.707) + 1.414 \cdot (-1.414)}{5 - 1}$$

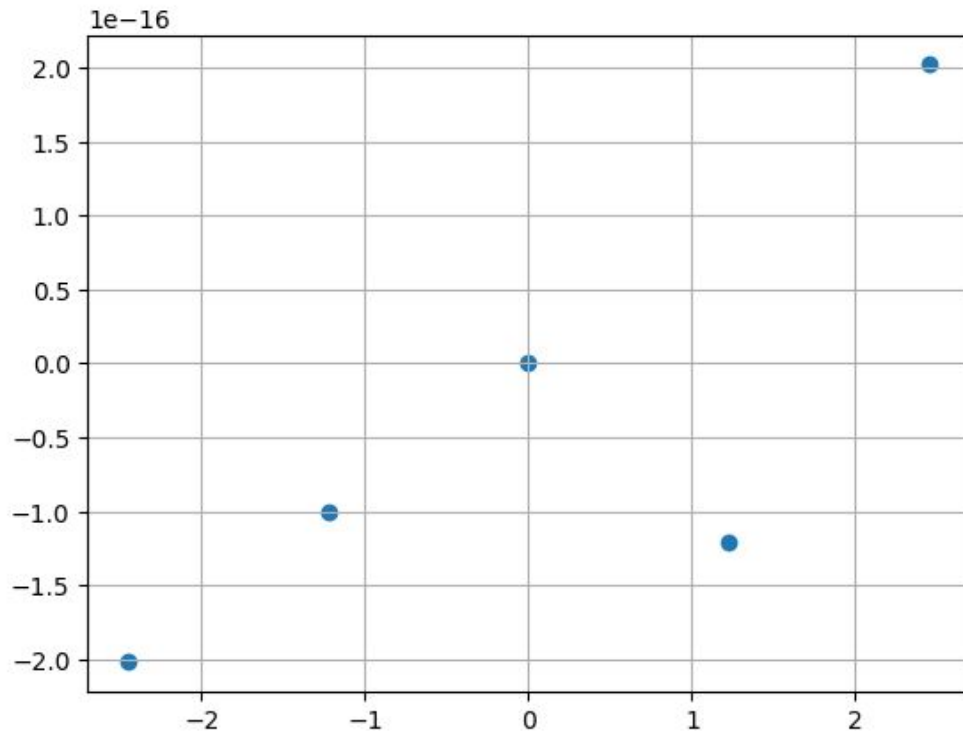
5 - 1

# Пример PCA (3)

Спроецированные данные  
 $X^* = XW$

-2.4	-2e-16
-1.2	-1e-16
0	0
1.2	-1.2e-16
2.4	2e-16

$$-1.414 \cdot 0.577 + 1.414 \cdot -0.577 - 1.414 \cdot 0.577$$



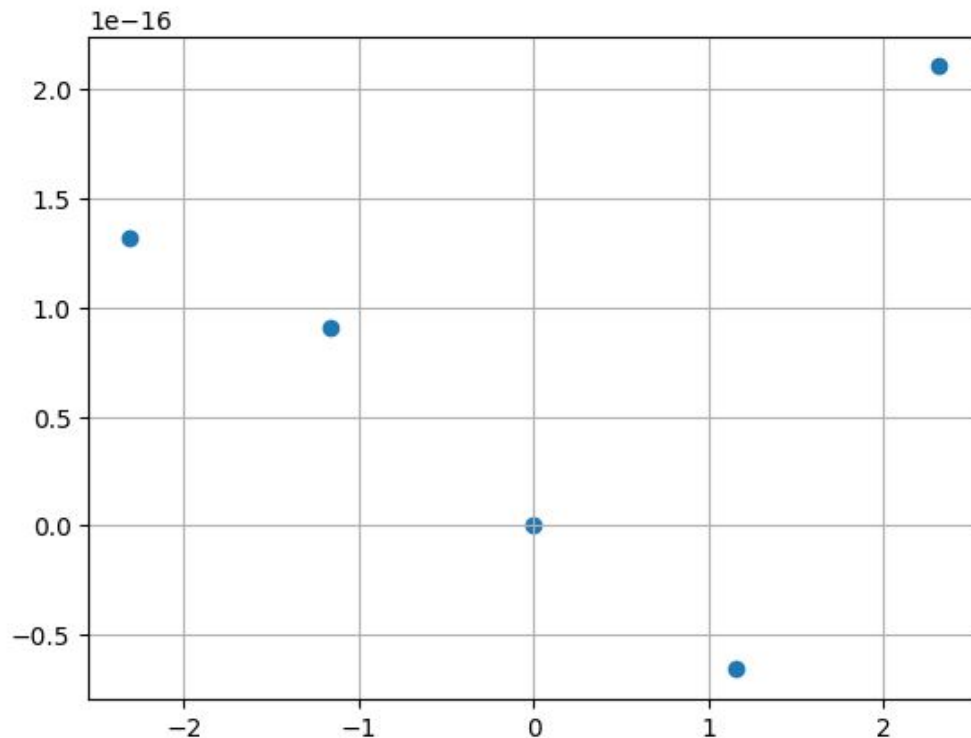


# PCA Sklearn

```
from sklearn.decomposition import PCA  
pca = PCA(n_components = 2)  
Xproj2 = pca.fit_transform(X)
```

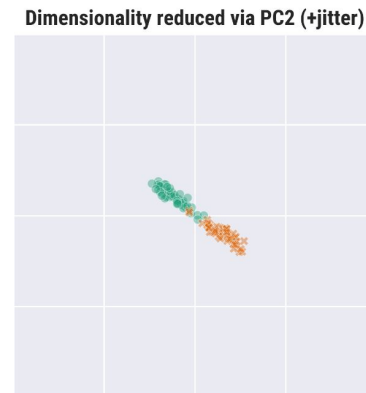
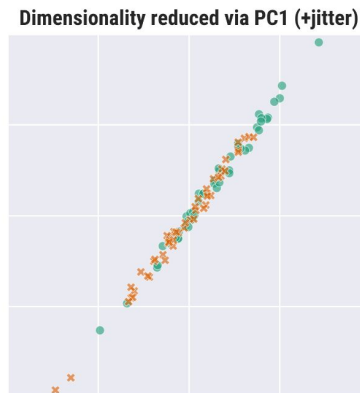
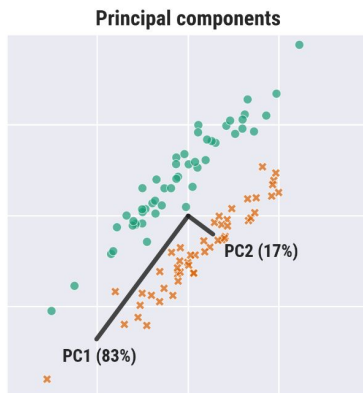
## Спроецированные данные

2.31	2.1e-16
1.15	-6.6e-17
0	0
-1.15	9e-17
-2.31	1.32e-16



# РСА для данных нескольких классов

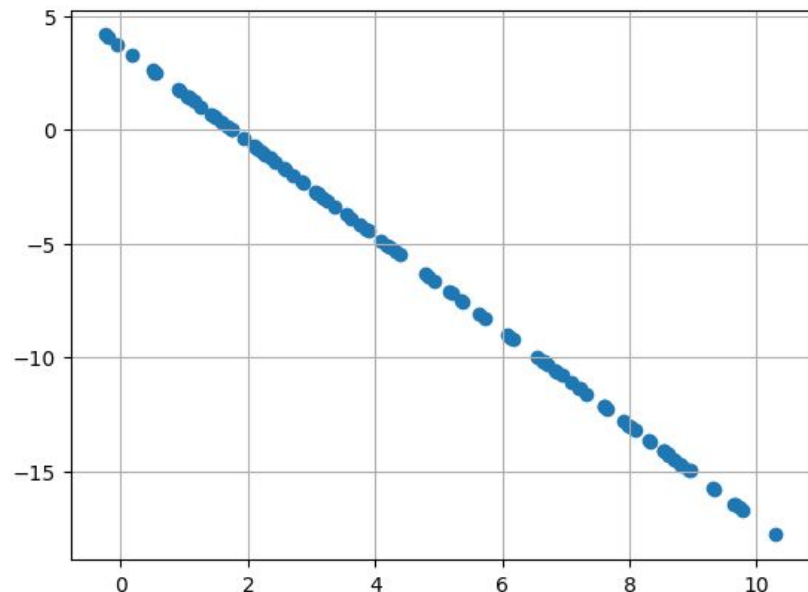
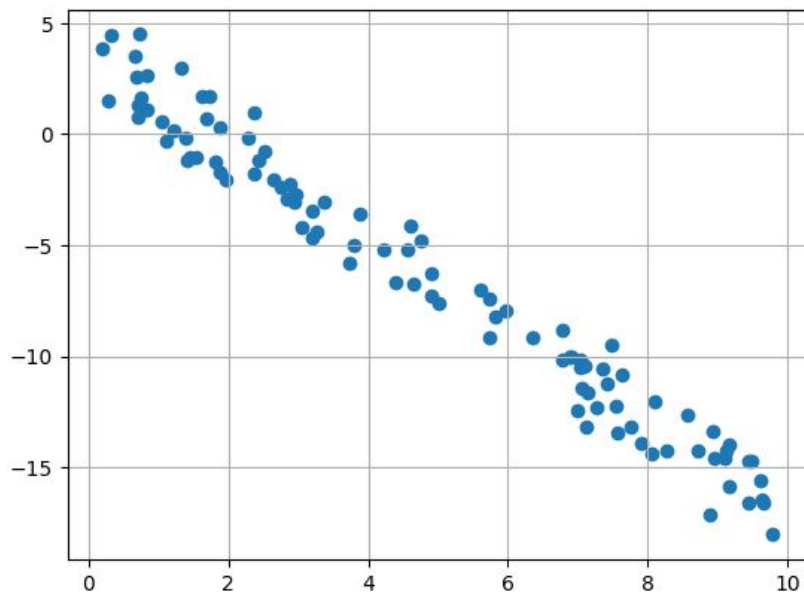
- РСА часто используется для того, чтобы сделать такую проекцию, в которой точки разных классов будут линейно разделимы.
- Линейная разделимость может не получиться, если данные имеют сложную структуру (например вложенные кольца). Тогда нужно формировать новые признаки или применять ядерный трюк (kernel trick)
- Также проблема, если разделимость не по направлению наибольшей дисперсии



# РСА для удаления шумов в данных

- Проводя понижение размерности, а затем восстанавливая данные обратно, можно удалить шумы.

*Преобразование данных из размерности 2 в 1 и их восстановление*



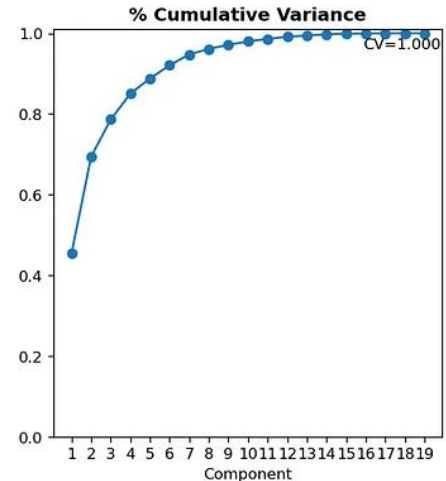
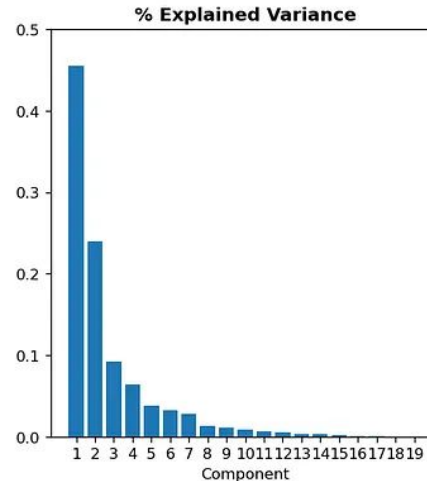
# Коэффициент объясненной дисперсии

- При снижении размерности пространства часть информации, для оценки “количества” сохраненной/потерянной информации необходимо анализировать собственные числа.
- Отношение собственного числа к сумме всех собственных чисел показывает % дисперсии, который он объясняет

- Остаточная дисперсия:  $\sum_{l=k+1}^n \lambda_l$

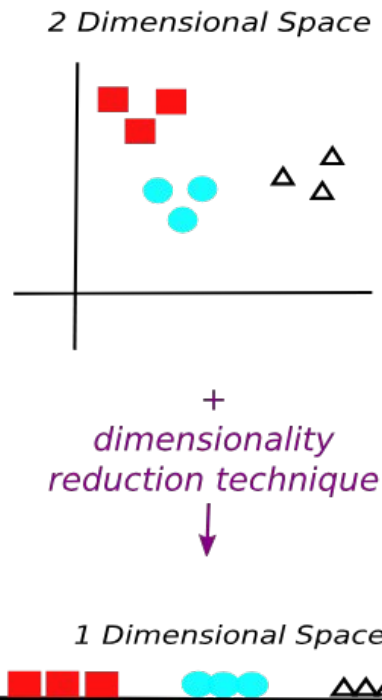
- Объясненная дисперсия:  $\sum_{l=1}^k \lambda_l$

- Отношение объясненной дисперсии ко всей дисперсии показывает % сохраненной информации



# t-SNE

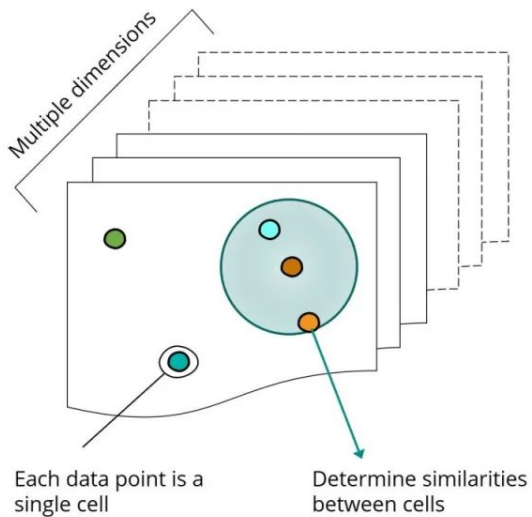
- Алгоритм понижения размерности данных для их визуализации.
- Хорошо работает, при очень больших размерностях
- Одностороннее нелинейное преобразование
- Упрощенно, алгоритм можно описать как то, что между каждой парой точек есть пружины. Сила натяжения пружины зависит от вероятности того, что пара точек близкие. Если отпустить пружины, то система придет в состояние равновесия.



Более сложно: <https://habr.com/ru/articles/267041/>

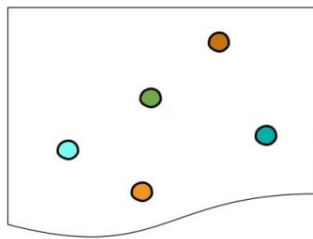
# t-SNE - алгоритм

## Stage 1

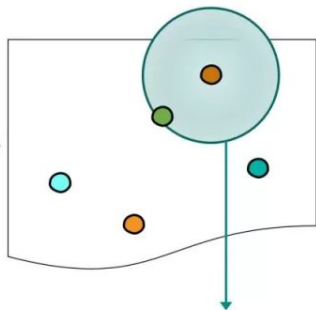


## Stage 2

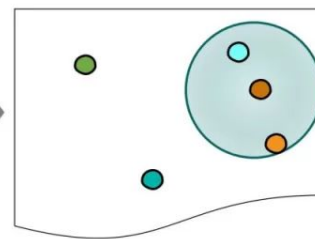
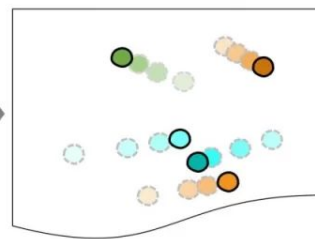
**a.** Randomly project cells as points on a low-dimensional plot



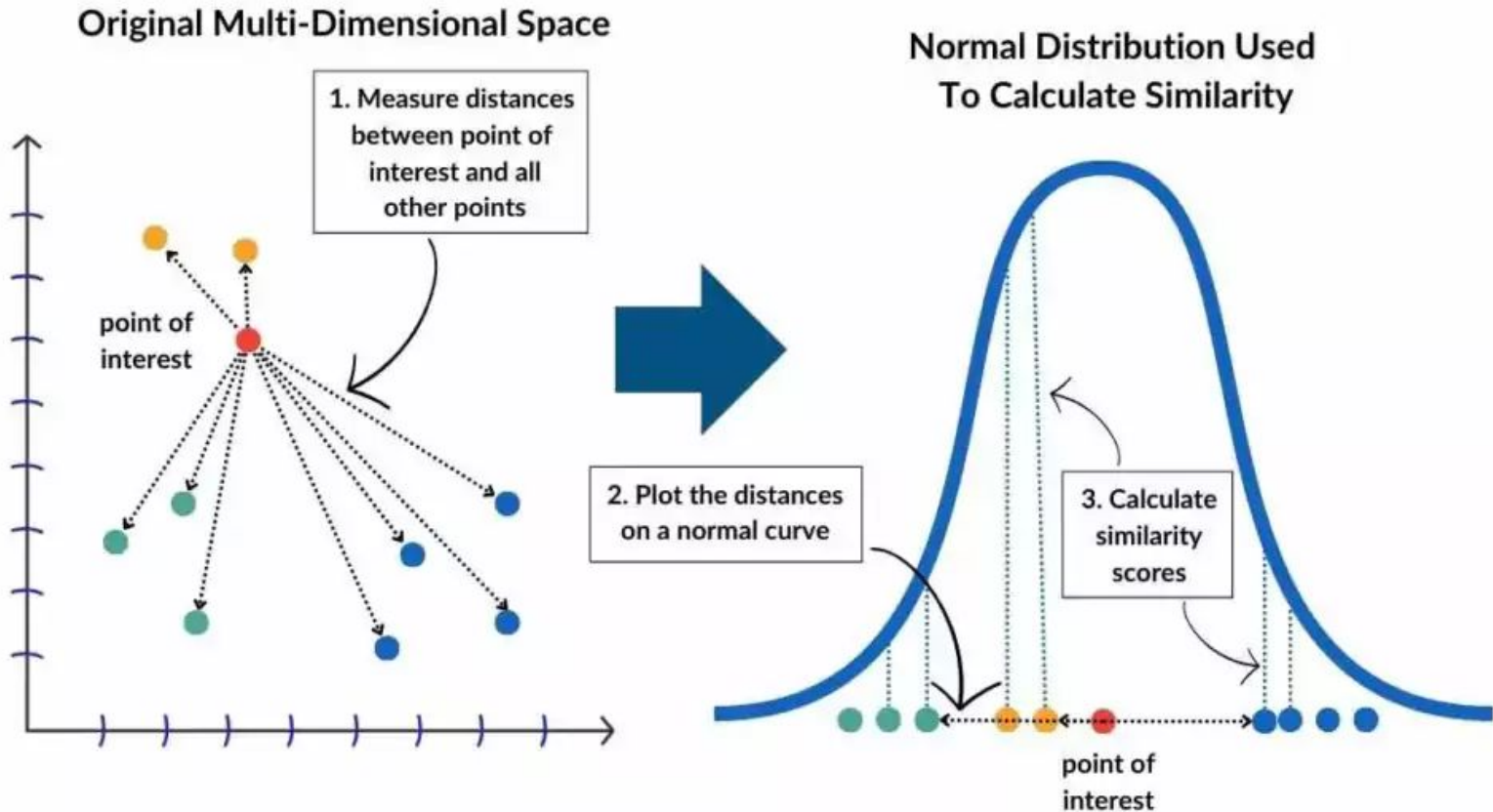
**b.** Determine similarities between points



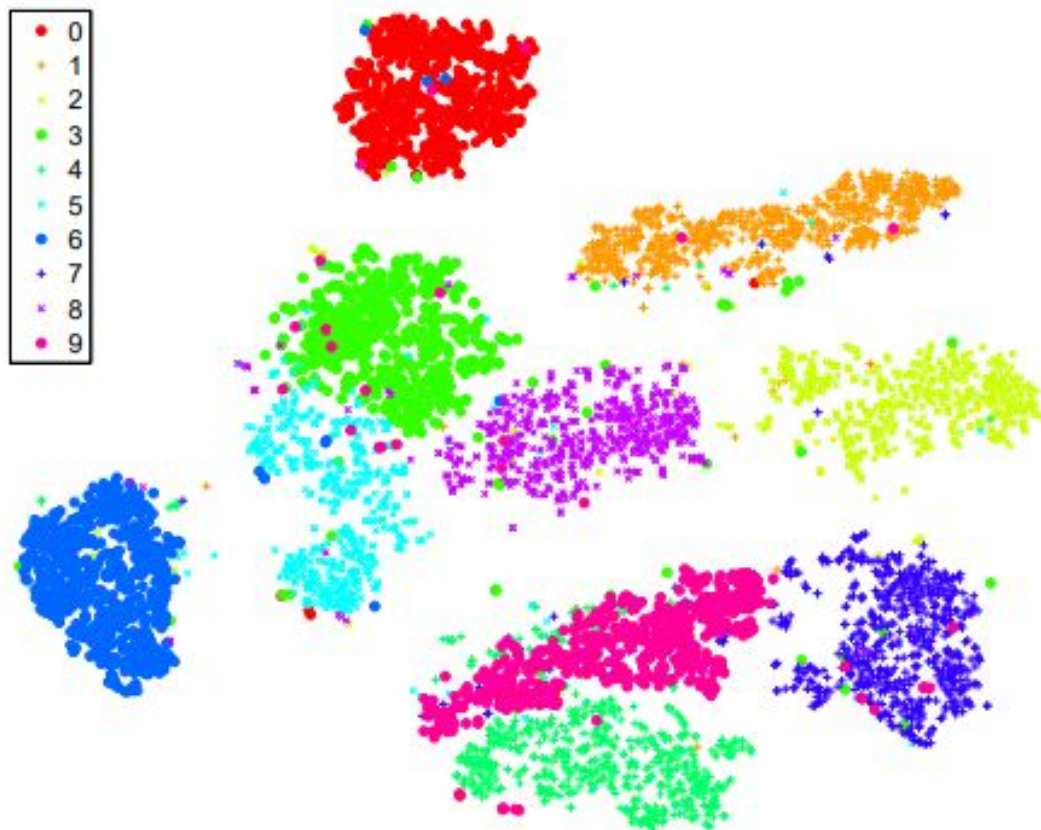
**c.** Move the points around until the similarities between points in low dimension resemble the similarities in high dimensions



# t-SNE - преобразование расстояний



# t-SNE пример на наборе MNIST





# t-SNE для текста

