**Оглавление**

[Основные понятия 2](#_Toc202341336)

[NumPy 3](#_Toc202341337)

[Массивы в NumPy 4](#_Toc202341338)

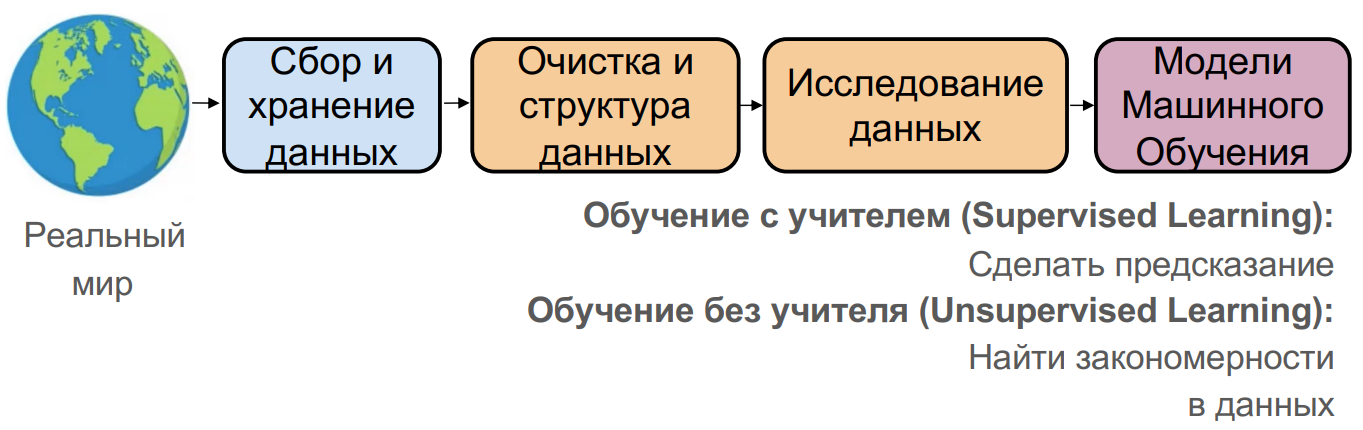
# Основные понятия

**Data Engineer** занимается сбором сырых данных, обработкой и хранением данных

**Data Analysis / Data Scientist** занимается очисткой и структурированием данных, исследованием данных

**Machine Learning Engineer** занимается исследованием данных и созданием Моделей Машинного Обучения

**Этапы работ по машинному обучению**



# NumPy

NumPy – это библиотека для языка программирования Python, предназначенная для работы с многомерными массивами и матрицами, а также для выполнения различных математических операций над ними.

Это первая и одна из основных библиотек в Python по машинному обучению. Очень многие другие библиотеки по машинному обучению основаны на NumPy

**Режим broadcast**

Важной особенностью библиотеки является возможность применить функции сразу ко многим элементам массива (без необходимости перебирать их в цикле – это как минимум быстрее)

NumPy использует **SIMD**-инструкции процессора (*Single Instruction Multiple Data*), которые позволяют выполнять одну и ту же операцию над несколькими значениями одновременно.

## Массивы в NumPy

### Способы создания массивов и матриц в numPy

|  |  |
| --- | --- |
| **np.array**  Преобразования списка Питон в массив numPy  Или сразу в матрицу  Причем может работать с матрицами более чем двух измерений | import numpy as np  my\_list = [1, 2, 3]  np\_list = np.array(my\_list)  print(np\_list)  *# [1 2 3]*  print(*type*(np\_list))  *# <class 'numpy.ndarray'>*  my\_matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  np\_matrix = np.array(my\_matrix)  print(np\_matrix)  *# [[1 2 3]*  *#  [4 5 6]*  *#  [7 8 9]]* |
| **np.arange**  Получение массива из диапазон numPy (верхняя граница *не включается* в диапазон) | np\_array = np.arange(0, 10, 2)  print(np\_array)  *# [0 2 4 6 8]* |
| **np.zeros**  Массив или матрица из 0 | np\_zeros\_array = np.zeros(5)  print(np\_zeros\_array)  *# [0. 0. 0. 0. 0.]*  np\_zeros\_matrix = np.zeros((3, 5), *int*)  print(np\_zeros\_matrix)  *# [[0 0 0 0 0]*  *#  [0 0 0 0 0]*  *#  [0 0 0 0 0]]* |
| **np.ones**  Массив или матрица из 1 | np\_ones\_array = np.ones(5)  print(np\_ones\_array)  *# [1. 1. 1. 1. 1.]* |
| **np.linspace**  Массив из чисел, *равностоящих* друг от друга, в заданном диапазоне (верхняя граница *включена* в диапазон) | array3 = np.linspace(0, 10, 3)  print(array3)  *# [ 0.  5. 10.]*  array10 = np.linspace(0, 10, 10)  array11 = np.linspace(0, 10, 11)  print(      array10  )  *# [ 0.  1.11111111  2.22222222  3.33333333  4.44444444  5.55555556  6.66666667  7.77777778  8.88888889  10 ]*  print(array11)  *# [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]* |
| **np.eye** создание единичной квадратной матрицы, где на главной диагонали – единицы, в остальных – нули | matrix = np.eye(5)  print(matrix)  *# [[1. 0. 0. 0. 0.]*  *#  [0. 1. 0. 0. 0.]*  *#  [0. 0. 1. 0. 0.]*  *#  [0. 0. 0. 1. 0.]*  *#  [0. 0. 0. 0. 1.]]* |

### Массивы и матрицы из случайных чисел

|  |  |
| --- | --- |
| **np.random.rand**  Массив или матрица из случайных чисел [0, 1). Количество строк и столбцов указывается не в кортеже  *Равномерное распределение* – вероятность получения любого числа от 0 до 1 | rand\_array = np.random.rand(3)  print(rand\_array)  *# [0.25271865 0.78779308 0.63061214]*  rand\_matrix = np.random.rand(3, 2)  print(rand\_matrix)  *# [[0.02027424 0.05326286]*  *#  [0.30627579 0.85974823]*  *#  [0.27939568 0.2943402 ]]* |
| **np.random.randn**  *Нормальное распределение (распределение Гаусса)* – математическое ожидание стремится к 0  То есть, чем ближе число к 0, тем выше вероятность, что функция вернет его | randn\_array = np.random.randn(10)  print(randn\_array)  *# [ 1.29392329 -0.86093267 -3.26004763 -0.8493568  -0.64312889 -0.1580054 1.13721477  0.73238211 -1.47633238 0.00989531]*  randn\_matrix = np.random.randn(2, 3)  print(randn\_matrix)  *# [[ 0.26856549 -0.66601058  1.88318487]*  *#  [-0.29179744 -0.70894104 -0.81990094]]* |
| **np.random.randint**  (начало диапазона, конец диапазона, размерность)  Массив или матрица с целыми случайными числами. Верхняя граница не включена | randn\_array = np.random.randint(0, 101, 5)  print(randn\_array)  *# [ 33  16  74 100  65]*  randn\_matrix = np.random.randint(0, 101, (4, 5))  print(randn\_matrix)  *# [[ 51  86  18  62  23]*  *#  [ 95  56  18 100  25]*  *#  [ 30  40  70  76  40]*  *#  [ 52  18  45  52  46]]* |
|  |  |
|  |  |
|  |  |

### Воспроизводимые наборы случайных чисел

Помогает, когда несколько раз надо воспроизвести те или иные случайные результаты.

Функция **seed(arg)** принимает аргумент – число, которому соответствует конкретный набор случайных чисел.

В следующий раз, когда мы вызовем seed с тем же аргументом, мы получим тот же набор случайных чисел (этот набор будет одинаковым при каждом запуске программы на разных машинах).

Традиционно, для описания набора используется число 42, но можно использовать любое

При использовании seed, функции *seed и rand должны вызываться в одном блоке кода*

np.random.seed(42)

rand\_array = np.random.rand(4)

print(rand\_array)  *# [0.37454012 0.95071431 0.73199394 0.59865848]*

rand\_array1 = np.random.rand(4)

print(rand\_array1)  *# [0.15601864 0.15599452 0.05808361 0.86617615]*

np.random.seed(42)

rand\_array2 = np.random.rand(5)

print(rand\_array2)  *# [0.37454012 0.95071431 0.73199394 0.59865848 0.15601864]*

### Работа с массивами и матрицами

|  |  |
| --- | --- |
| **\_.reshape**  Переформатирование матрицы, массива  Количество элементов в новом массиве / матрице должно совпадать с количеством элементов в исходном массиве | array = np.arange(0, 10)  print(array)  *# [0 1 2 3 4 5 6 7 8 9]*  reshaped\_array = array.reshape(2, 5)  print(reshaped\_array)  *# [[0 1 2 3 4]*  *#  [5 6 7 8 9]]* |
| **\_.shape**  Посмотреть размерность массива / матрицы | array = np.random.randint(0, 101, 10)  print(array.shape)  *# (10,)*  reshaped = array.reshape(2, 5)  print(reshaped.shape)  *# (2, 5)* |
| **\_.min**  Минимальное число в массиве | rand\_array = np.random.randint(0, 101, 10)  print(rand\_array)  *# [69 15 78 80 65  2 18 35 29 37]*  min\_value = rand\_array.min()  print(min\_value)  *# 2* |
| **\_.max**  Максимальное число в массиве | rand\_array = np.random.randint(0, 101, 10)  print(rand\_array)  *# [69 15 78 80 65  2 18 35 29 37]*  max\_value = rand\_array.max()  print(max\_value)  *# 80* |
| \_.**argmin**  \_.**argmax**  Индекс минимальных/ максимальных чисел в массиве | rand\_array = np.random.randint(0, 101, 10)  print(rand\_array)  *# [69 15 78 80 65  2 18 35 29 37]*  index\_min = rand\_array.argmin()  index\_max = rand\_array.argmax()  print(index\_min)  *# 5*  print(index\_max)  *# 3* |
| \_.**dtype**  Посмотреть тип данных в массиве | rand\_array\_int = np.random.randint(0, 101, 10)  print(rand\_array\_int.dtype)  *# int32*  rand\_array\_float = np.random.rand(0, 101, 10)  print(rand\_array\_float.dtype)  *# float64* |