# Библиотека NumPy Python

расскажи про библиотеку numpy python

NumPy (Numerical Python) — это одна из ключевых библиотек для научных вычислений в Python. Она предоставляет высокопроизводительные структуры данных и функции для работы с многомерными массивами и матрицами, а также широкий набор математических инструментов.

---

## ✅ Основные особенности NumPy

1. **Массивы (ndarray)**
   Основной объект в NumPy — это `ndarray` (N-dimensional array), который:
   - хранит элементы одного типа (в отличие от списков Python);
   - имеет фиксированный размер (в отличие от динамических списков);
   - поддерживает векторизацию (операции выполняются сразу на всем массиве, без циклов).
2. **Высокая скорость**
   NumPy написан на C, что делает операции с массивами значительно быстрее, чем с обычными списками Python.
3. **Математические функции**
   Есть встроенные функции для:
   - линейной алгебры ( `numpy.linalg` );
   - статистики ( `mean` , `std` , `var` );
   - тригонометрии, экспонент, логарифмов и т.д.
4. **Бродкастинг (Broadcasting)**
   Возможность выполнять операции над массивами разной формы без явных циклов.
5. **Интеграция**
   NumPy — основа для других библиотек: **pandas**, **scikit-learn**, **TensorFlow**, **PyTorch** и многих других.

---

## ✅ Установка

bash

```
pip install numpy
```

---

## ✅ Базовые примеры

```python
import numpy as np

# Создание массива
a = np.array([1, 2, 3, 4])
print(a)  # [1 2 3 4]

# Двумерный массив
b = np.array([[1, 2], [3, 4]])
print(b.shape)  # (2, 2)

# Операции с массивами
print(a + 10)  # [11 12 13 14]
print(a * 2)   # [2 4 6 8]

# Статистика
print(a.mean())  # 2.5

# Специальные массивы
zeros = np.zeros((3, 3))      # матрица из нулей
ones = np.ones((2, 4))        # матрица из единиц
rand = np.random.rand(2, 3)   # случайные числа
```

---

## ✅ Популярные функции

- `np.arange(0, 10, 2)` → `[0 2 4 6 8]`
- `np.linspace(0, 1, 5)` → `[0. 0.25 0.5 0.75 1.]`
- `np.reshape(a, (2, 2))` → меняет форму массива
- `np.dot(A, B)` → матричное умножение
- `np.sum(a, axis=0)` → сумма по оси

---

Хочешь, я сделаю **таблицу с основными функциями NumPy для практики** или **покажу, как NumPy отличается от обычных списков по скорости**?

приведи основные функции

Вот список основных функций и методов NumPy, сгруппированных по категориям, чтобы было удобно ориентироваться:

---

## ✅ 1. Создание массивов

| Функция | Описание |
|---|---|
| `np.array([1,2,3])` | Создать массив из списка |
| `np.zeros((3,3))` | Массив из нулей |
| `np.ones((2,4))` | Массив из единиц |
| `np.full((2,3), 7)` | Массив, заполненный 7 |
| `np.eye(3)` | Единичная матрица (3×3) |
| `np.arange(0, 10, 2)` | Последовательность чисел с шагом |
| `np.linspace(0, 1, 5)` | Равномерное разбиение отрезка |
| `np.random.rand(2, 3)` | Случайные числа (0–1) |
| `np.random.randint(0,10,(3,3))` | Случайные целые |

---

## ✅ 2. Основные операции

| Функция | Описание |
| --- | --- |
| `a + b`, `a - b`, `a * b` | Поэлементные операции |
| `a / b`, `a ** 2` | Деление, возведение в степень |
| `np.add(a,b)`, `np.subtract(a,b)` | Альтернатива операторам |
| `np.multiply(a,b)`, `np.divide(a,b)` | Умножение, деление |

---

## ✅ 3. Статистические функции

| Функция | Описание |
| --- | --- |
| `np.sum(a)` | Сумма элементов |
| `np.mean(a)` | Среднее |
| `np.median(a)` | Медиана |
| `np.std(a)` | Стандартное отклонение |
| `np.var(a)` | Дисперсия |
| `np.min(a)`, `np.max(a)` | Минимум, максимум |
| `np.argmin(a)`, `np.argmax(a)` | Индекс минимума/максимума |

---

## ✅ 4. Работа с формой массива

| Функция | Описание |
| --- | --- |
| `a.shape` | Размерность |

| Функция | Описание |
| --- | --- |
| `a.reshape(2,3)` | Изменить форму |
| `a.flatten()` | Превратить в 1D |
| `a.T` | Транспонирование |
| `np.vstack((a,b))` | Склеить по вертикали |
| `np.hstack((a,b))` | Склеить по горизонтали |

---

## ✅ 5. Линейная алгебра

| Функция | Описание |
| --- | --- |
| `np.dot(a,b)` | Матричное умножение |
| `np.matmul(a,b)` | То же самое (современный вариант) |
| `np.linalg.inv(a)` | Обратная матрица |
| `np.linalg.det(a)` | Определитель |
| `np.linalg.eig(a)` | Собственные значения/векторы |

---

## ✅ 6. Логические операции

| Функция | Описание |
| --- | --- |
| `np.where(a > 5)` | Индексы элементов > 5 |
| `np.all(a > 0)` | Все ли > 0 |

| Функция | Описание |
| --- | --- |
| `np.any(a < 0)` | Есть ли < 0 |
| `(a > 3) & (a < 7)` | Логические маски |

---

Хочешь, сделаю **PDF-шпаргалку с примерами кода для каждой функции** или **таблицу с объяснением + примерами кода прямо здесь**?

напиши применение в datascience и ml

NumPy — это фундамент для работы с данными в **Data Science** и **Machine Learning**. Его применение охватывает все этапы: от загрузки данных до построения моделей. Вот основные области:

---

## ✅ 1. Хранение и обработка данных

- **Почему NumPy:** массивы `ndarray` занимают меньше памяти, чем списки, и позволяют быстро выполнять операции.
- **Примеры:**

```python
import numpy as np

# Загрузка данных из CSV
data = np.genfromtxt('data.csv', delimiter=',')

# Нормализация признаков
data = (data - np.mean(data, axis=0)) / np.std(data, axis=0)
```

---

## ✅ 2. Предобработка данных

- Масштабирование и нормализация признаков перед обучением:

```python
```

```
X = np.array([[1, 200], [2, 300], [3, 400]])
X_scaled = (X - X.mean(axis=0)) / X.std(axis=0)
```

- Заполнение пропусков:

```
python

X[np.isnan(X)] = np.nanmean(X)
```

---

## ✅ 3. Работа с изображениями

- Изображения → это просто массивы пикселей.
- Пример: изменить размерность RGB-картинки (H×W×3):

```
python

image = np.random.randint(0, 255, (256, 256, 3))
flat = image.reshape(-1, 3)  # векторы пикселей
```

---

## ✅ 4. Генерация данных

- Для тестирования моделей нужны синтетические данные:

```
python

# Случайные признаки
X = np.random.rand(100, 5)
# Случайная целевая переменная
y = np.random.randint(0, 2, 100)
```

---

## ✅ 5. Линейная алгебра для ML

- В машинном обучении математика важна:
  - **Градиентный спуск** → операции над матрицами
  - **Регрессия** → умножения матриц, транспонирование

```
python
```

```
X = np.array([[1, 2], [3, 4], [5, 6]])
y = np.array([7, 8, 9])

# Нормальное уравнение для линейной регрессии:
theta = np.linalg.inv(X.T @ X) @ X.T @ y
```

---

## ✅ 6. Основы нейросетей

- **Forward pass** и **backpropagation** считаются через NumPy:

```
python

# Сигмоида и предсказание
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

X = np.random.rand(4, 3)
W = np.random.rand(3, 1)
b = 0.5
y_pred = sigmoid(X @ W + b)
```

---

## ✅ 7. Оптимизация и векторизация

- Вместо медленных Python-циклов:

```
python

# Плохо (медленно):
result = []
for i in range(1000000):
    result.append(i**2)

# Быстро:
arr = np.arange(1000000)**2
```

---

### Где именно NumPy в DS/ML?

✔ Под капотом **pandas**, **scikit-learn**, **TensorFlow**, **PyTorch**
✔ В алгоритмах: линейная регрессия, PCA, k-means, градиентный спуск
✔ В подготовке данных: нормализация, кодирование, заполнение пропусков

---

Хочешь, я сделаю **таблицу “NumPy + типичная задача Data Science” с кодом** или **разбор, как реализовать линейную регрессию полностью на NumPy без sklearn**?