Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант 12

Выполнил:
Рябинин Егор Алексеевич
2 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии
Воронкин Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г

Тема: Основы работы с библиотекой NumPy

Цель: исследовать базовые возможности библиотеки NumPy языка программирования Python.

Порядок выполнения работы:

Ссылка на репозиторий GitHib:

https://github.com/EgorGorilla/Lab2_Artificial-Intelligence-and-Machine-Learning

Задание 1. Создание и изменение массивов

Создайте массив NumPy размером 3×3, содержащий числа от 1 до 9. Умножьте все элементы массива на 2, а затем замените все элементы больше 10 на 0. Выведите итоговый массив.

In [68]:
```
import numpy as np
A = np.array([[1,2,3],
              [4,5,6],
              [7,8,9]])
A=A*2
for i in range(3):
    for j in range(3):
        if (A[i][j]>10):
            A[i][j]=0
A
```

Рисунок 1 – Задание 1. Создание и изменение массивов

```
Out[68]: array([[ 2,  4,  6],
                [ 8, 10,  0],
                [ 0,  0,  0]])
```

Рисунок 2 – Результат работы программы к заданию 1

Задание 2. Работа с булевыми масками

Создайте массив NumPy из 20 случайных целых чисел от 1 до 100. Найдите и выведите все элементы, которые делятся на 5 без остатка. Затем замените их на -1 и выведите обновленный массив.

In [190…
```
A = np.random.randint(1, 101, size=(20))
print("Исходный массив: ",A)
print("Числа, которые делятся на 5 без остатка:")
for i in range(20):
    if (A[i]%5==0):
        print(A[i])
        A[i]=-1
print("Массив, в котором элементы, которые делятся на 5 без остатка, заменены на -1: ",A)
```

Рисунок 3 – Задание 2. Работа с булевыми масками

```
Исходный массив:  [18  6 22 88  3 50 42 15 94 47 67 75 19 40 72 78 59 91 32 80]
Числа, которые делятся на 5 без остатка:
50
15
75
40
80
Массив, в котором элементы, которые делятся на 5 без остатка, заменены на -1:  [18  6 22 88  3 -1 42 -1 94 47
67 -1 19 -1 72 78 59 91 32 -1]
```

Рисунок 4 – Результат работы программы к заданию 2

Задание 3. Объединение и разбиение массивов.

Создайте два массива NumPy размером 1×5, заполненные случайными числамиот 0 до 50.

- Объедините эти массивы в один двумерный массив (по строкам).
- Разделите полученный массив на два массива, каждый из которых содержит 5 элементов.

Выведите все промежуточные и итоговые результаты

In [234…

```
A = np.random.randint(0,51, size=(1,5))
B = np.random.randint(0,51, size=(1,5))
print("Массивы 1x5, заполненные случайными числами от 0 до 50:")
print("A =",A);print("B =", B)
matrix = np.vstack((A,B))
print("Массивы A и B, обьединенные в один двумерный массив: ")
print(matrix)
A_new, B_new = np.vsplit(matrix, 2)
print("Двумерный массив, который был разделен обратно на два массива: ")
print("A =",A_new);print("B =", B_new)
```

Рисунок 5 – Задание 3. Объединение и разбиение массивов

```
Массивы 1x5, заполненные случайными числами от 0 до 50:
A = [[25 19 42 35 15]]
B = [[ 8  7 14 32 35]]
Массивы A и B, обьединенные в один двумерный массив:
[[25 19 42 35 15]
 [ 8  7 14 32 35]]
Двумерный массив, который был разделен обратно на два массива:
A = [[25 19 42 35 15]]
B = [[ 8  7 14 32 35]]
```

Рисунок 6 – Результат работы программы к заданию 3

Задание 4. Генерация и работа с линейными последовательностями

Создайте массив из 50 чисел, равномерно распределенных от -10 до 10. Вычислите сумму всех элементов, сумму положительных элементов и сумму отрицательных элементов. Выведите результаты.

In [286…

```
A = np.linspace(-10, 10, 50)
C = np.sum(A)
print("Массив из 50 чисел, равномерно распределенных от -10 до 10:\n", A)
print("Сумма всех элементов массива: ", C)
B = np.sum(A[A>0])
print("Сумма всех положительных элементов массива: ", B)
D = np.sum(A[A<0])
print("Сумма всех отрицательных элементов массива: ", D)
```

Рисунок 7 – Задание 4. Генерация и работа с линейными последовательностями

```
Массив из 50 чисел, равномерно распределенных от -10 до 10:
 [-10.          -9.59183673  -9.18367347  -8.7755102   -8.36734694
  -7.95918367  -7.55102041  -7.14285714  -6.73469388  -6.32653061
  -5.91836735  -5.51020408  -5.10204082  -4.69387755  -4.28571429
  -3.87755102  -3.46938776  -3.06122449  -2.65306122  -2.24489796
  -1.83673469  -1.42857143  -1.02040816  -0.6122449   -0.20408163
   0.20408163   0.6122449    1.02040816   1.42857143   1.83673469
   2.24489796   2.65306122   3.06122449   3.46938776   3.87755102
   4.28571429   4.69387755   5.10204082   5.51020408   5.91836735
   6.32653061   6.73469388   7.14285714   7.55102041   7.95918367
   8.36734694   8.7755102    9.18367347   9.59183673  10.        ]
Сумма всех элементов массива:  7.105427357601002e-15
Сумма всех положительных элементов массива:  127.55102040816328
Сумма всех отрицательных элементов массива:  -127.55102040816327
```

Рисунок 8 – Результат работы программы к заданию 4

Задание 5. Работа с диагональными и единичными матрицами

Создайте:

- Единичную матрицу размером 4x4.
- Диагональную матрицу размером 4x4 с диагональными элементами (5,10,15,20), не используя циклы.

Найдите сумму всех элементов каждой из этих матриц и сравните результаты

In [314…

```
A = np.eye(4)
print("Единичная матрица размером 4 порядка:\n",A)
B = np.diag([5, 10, 15, 20])
print("Диагональная матрица 4 порядка с диагональными элементами (5,10,15,20):\n", B)
k = np.sum(A)
k1 = np.sum(B)
print("Сумма всех элементов единичной матрицы:",k)
print("Сумма всех элементов диагональной матрицы:",k1)
```

Рисунок 9 – Задание 5. Работа с диагональными и единичными матрицами

```
Единичная матрица размером 4 порядка:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
Диагональная матрица 4 порядка с диагональными элементами (5,10,15,20):
 [[ 5  0  0  0]
 [ 0 10  0  0]
 [ 0  0 15  0]
 [ 0  0  0 20]]
Сумма всех элементов единичной матрицы: 4.0
Сумма всех элементов диагональной матрицы: 50
```

Рисунок 10 – Результат работы программы к заданию 5

Задание 6. Создание и базовые операции с матрицами

Создайте две квадратные матрицы NumPy размером 3x3, заполненые случайными целыми числами от 1 до 20.
Вычислите и выведите:

- Их сумму
- Их разность
- Их поэлементное произведение

In [356…

```
A = np.random.randint(1,21, size=(3,3))
B = np.random.randint(1,21, size=(3,3))
print("A = \n",A)
print("B = \n",B)
C = A + B
print("A + B =\n",C)
D = A - B
print("A - B =\n",D)
E = A*B
print("A * B (поэлементное произведение) =\n",E)
```

Рисунок 11 – Задание 6. Создание и базовые операции с матрицами

```
A =
 [[ 8 15 16]
 [ 6 12  9]
 [16 17 14]]
B =
 [[ 1 13 14]
 [ 2  8 11]
 [19 18 17]]
A + B =
 [[ 9 28 30]
 [ 8 20 20]
 [35 35 31]]
A - B =
 [[ 7  2  2]
 [ 4  4 -2]
 [-3 -1 -3]]
A * B (поэлементное произведение) =
 [[  8 195 224]
 [ 12  96  99]
 [304 306 238]]
```

Рисунок 12 – Результат работы программы к заданию 6

Задание 7. Умножение матриц

Создайте две матрицы NumPy:

- Первую размером 2x3, заполненную случайными числами от 1 до 10.
- Вторую размеров 3x2, заполненную случайными числами от 1 до 10.

Выполните матричное умножение и выведите результат.

In [362…

```
A = np.random.randint(1,11, size=(2,3))
B = np.random.randint(1,11, size=(3,2))
print("A = \n",A)
print("B = \n",B)
C = np.dot(A,B)
print("Произведение матрицы A на матрицу B:\n",C)
```

Рисунок 13 – Задание 7. Умножение матриц

```
A =
 [[ 5  7  4]
 [ 8  3 10]]
B =
 [[9 8]
 [8 2]
 [5 2]]
Произведение матрицы A на матрицу B:
 [[121  62]
 [146  90]]
```

Рисунок 14 – Результат работы программы к заданию 7

Задание 8. Определитель и обратная матрица

Создайте случайную квадратную матрицу 3x3. Найдите и выведите:

- Определитель этой матрицы
- Обратную матрицу (если существует, иначе выведите сообщение, что матрица вырождена)

Используйте функции np.linalg.det и np.linalg.inv

In [469…

```
A = np.random.rand(3, 3)
print("Матрица A =\n",A)
C = np.linalg.det(A)
print("det A = ", C)
D = np.linalg.inv(A)
if (C != 0):
    print("A⁻¹ = \n", D)
else:
    print("det A = 0, обратной матрицы не существует!")
```

Рисунок 15 – Задание 8. Определитель и обратная матрица

```
Матрица A =
 [[0.50650186 0.62886753 0.07887878]
 [0.20213945 0.65439707 0.82874358]
 [0.77868256 0.71258994 0.17992485]]
det A =  0.11464172628121103
A⁻¹ =
 [[-4.12425788 -0.49668365  4.09582006]
 [ 5.31183787  0.25916168 -3.52241429]
 [-3.18841194  1.12315165  1.78237372]]
```

Рисунок 16 – Результат работы программы к заданию 8

Задание 9. Транспонирование и след матрицы

Создайте матрицу NumPy размером 4x4, содержащую случайные целые числа от 1 до 50. Выведите:

- Исходную матрицу
- Транспонированную матрицу
- След матрицы (сумму элементов на главной диагонали)

Используйте np.trace для нахождения следа.

In [486…

```
A = np.random.randint(1,51, size=(4,4))
print("Исходная матрица: \n",A)
print("Транспонированная матрица: \n", np.transpose(A))
print("След матрицы: ", np.trace(A))
```

Рисунок 17 – Задание 9. Транспонирование и след матрицы

```
Исходная матрица:
 [[47  5  3  6]
 [ 9 50 32 30]
 [41 46  4  7]
 [29 20 15 44]]
Транспонированная матрица:
 [[47  9 41 29]
 [ 5 50 46 20]
 [ 3 32  4 15]
 [ 6 30  7 44]]
След матрицы:  145
```

Рисунок 18 – Результат работы программы к заданию 9

Задание 10. Системы линейных уравнений

Решите систему линейных уравнений вида: $\begin{cases} 2x + 3y - z = 5 \\ 4x - y + 2x = 6 \\ -3x + 5y + 4z = -2 \end{cases}$

Используйте матричное представление AX = B, где A - матрица коэффициентов, x - вектор неизвестных, B - вектор правой части. Решите систему с помощью np.linalg.solve и выведите результат

In [525…

```
A = np.array([[2,3,-1],
              [4,-1,2],
              [-3,5,4]])
B = np.array([[5],[6],[-2]])
print("Ответ:\n",np.linalg.solve(A,B))
```

Рисунок 19 – Задание 10. Системы линейных уравнений

```
Ответ:
 [[1.63963964]
 [0.57657658]
 [0.00900901]]
```

Рисунок 20 – Результат выполнения программы к заданию 10

Индивидуальное задание.

Задание предусматривает построение СЛУ. Решите полученную СУ с использованием библиотеки NumPy.
Для решения системы используйте **метод Крамера** и **матричный метод**. Сравните полученные результаты, с результатами, полученными с помощью **np.linalg.solve**.

Вариант 12.
На концерт продано 500 билетов трех категорий: стандартные, премиум и VIP.
Билеты премиум стоят в 1.5 раза дороже стандартных, а VIP — в 2 раза дороже премиум.
Общий доход от продажи составил 600 000 рублей.
Сколько билетов каждого типа было продано, если известно, что стандартных билетов было на 100 больше, чем премиум?

upd: т.к у нас не указана цена стандартного билета, то примем его значение за 1000

Получим систему уравнений: $\begin{cases} s + 1,5p + 3v = 600 \\ s + p + v = 500 \\ s - p = 100 \end{cases}$

Рисунок 21 – Индивидуальное задание

```
A = np.array([[1,1.5,3],
              [1,1,1],
              [1,-1,0]])
B = np.array([[600],
              [500],
              [100]])
G = np.linalg.inv(A)
matr = G@B
numpy = np.linalg.solve(A,B)
print("Ответ к СЛУ с помощью np.linalg.solve:\n",numpy)
print("Ответ к СЛУ с помощью матричного метода:\n",matr)
det = np.linalg.det(A)
s1 = np.array([[600,1.5,3],
               [500,1,1],
               [100,-1,0]])
p1 = np.array([[1,600,3],
               [1,500,1],
               [1,100,0]])
v1 = np.array([[1,1.5,600],
               [1,1,500],
               [1,-1,100]])
dets = np.linalg.det(s1)
detp = np.linalg.det(p1)
detv = np.linalg.det(v1)
s = dets/det
p = detp/det
v = detv/det
kramer = np.array([[s],
               [p],
               [v]])
print("Ответ к СЛУ с помощью метода Крамера:\n", kramer)
```

Рисунок 22 – Код для индивидуального задания

```
Ответ к СЛУ с помощью np.linalg.solve:
 [[ 3.00000000e+02]
 [ 2.00000000e+02]
 [-3.96508223e-15]]
Ответ к СЛУ с помощью матричного метода:
 [[3.0000000e+02]
 [2.0000000e+02]
 [1.0658141e-14]]
Ответ к СЛУ с помощью метода Крамера:
 [[ 3.00000000e+02]
 [ 2.00000000e+02]
 [-3.96508223e-15]]
```

Рисунок 23 – Результат выполнения программы

Контрольные вопросы:

1. Назначение библиотеки NumPy

NumPy — это библиотека для работы с многомерными массивами, матрицами и числами в Python. Она предоставляет высокопроизводительные структуры данных и операции для численных вычислений, включая функции для математических, логических, статистических и алгебраических операций.

2. Массивы ndarray

Массивы ndarray (N-dimensional array) — это основная структура данных библиотеки NumPy. Они представляют собой многомерные таблицы, где каждый элемент имеет одинаковый тип данных. Массивы могут быть одномерными, двумерными и многомерными.

3. Доступ к частям многомерного массива

Доступ к частям массива осуществляется с помощью индексации. Можно использовать:

- Индексацию с помощью числовых индексов для одномерных и многомерных массивов.
- Срезы (например, arr[1:3, 2:4] для двумерных массивов).
- Логическую индексацию или маски.

4. Расчет статистик по данным

NumPy предоставляет функции для расчета различных статистик, таких как:

- Среднее значение: np.mean()
- Медиана: np.median()
- Стандартное отклонение: np.std()
- Минимум и максимум: np.min(), np.max()
- Квантиль: np.percentile()
- Корреляция: np.corrcoef()

5. Выборка данных из массивов `ndarray`

Выборка данных из массива осуществляется через индексацию, срезы или логическую маску. Например, чтобы выбрать все элементы, которые больше 5:

```
arr[arr > 5]
```

6. Основные виды матриц и векторов

- Вектор: одномерный массив. Создается с помощью `np.array([1, 2, 3])` или `np.arange()`.
- Матрица: двумерный массив. Создается с помощью `np.array([[1, 2], [3, 4]])` или `np.zeros((2, 2))`.
- Единичная матрица: `np.eye(n)`
- Нулевая матрица: `np.zeros((n, m))`
- Матрица с произвольными числами: `np.random.rand(n, m)`

7. Транспонирование матриц

Транспонирование матрицы меняет строки на столбцы. В NumPy это делается через метод `.T`:

A.T

8. Свойства операции транспонирования матриц

- Транспонирование единичной матрицы дает единичную матрицу.
- Транспонирование транспонированной матрицы возвращает исходную матрицу: `(A.T).T = A`.
- Транспонирование произведения матриц: `(A * B).T = B.T * A.T`.

9. Средства NumPy для транспонирования матриц

Для транспонирования в NumPy можно использовать:

- `.T`
- `np.transpose(A)`

10. Основные действия над матрицами

Основные действия:

- Сложение и вычитание: `A + B`, `A - B`
- Умножение: `A * B`, или `np.dot(A, B)` для матричного умножения
- Транспонирование: `A.T`
- Определитель: `np.linalg.det(A)`

- Обратная матрица: `np.linalg.inv(A)`

11. Умножение матрицы на число

Для умножения матрицы на число используется стандартная операция умножения:

A * c

12. Свойства операции умножения матрицы на число

- Это дистрибутивная операция: `(c * A) + (c * B) = c * (A + B)`.
- Умножение на 1 оставляет матрицу неизменной: `1 * A = A`.
- Умножение на 0 дает нулевую матрицу: `0 * A = 0`.

13. Сложение и вычитание матриц

Сложение и вычитание матриц выполняется через стандартные операторы:

A + B

A - B

14. Свойства операций сложения и вычитания матриц

- Операции ассоциативны и коммутативны: `A + B = B + A`, `(A + B) + C = A + (B + C)`.
- Вычитание матриц не является коммутативным: `A - B ≠ B - A`.

15. Средства в NumPy для сложения и вычитания матриц

Для выполнения этих операций в NumPy можно использовать обычные операторы:

A + B

A – B

16. Операция умножения матриц

Для матричного умножения в NumPy используется функция `np.dot()` или оператор `@`:

np.dot(A, B)

A @ B

17. Свойства операции умножения матриц

- Умножение не является коммутативным: `A * B ≠ B * A`.
- Ассоциативность: `(A * B) * C = A * (B * C)`.
- Дистрибутивность: `A * (B + C) = A * B + A * C`.

18. Средства NumPy для умножения матриц

Для умножения матриц можно использовать:

- `np.dot(A, B)`
- `A @ B`
- `np.matmul(A, B)`

19. Определитель матрицы

Определитель матрицы — это скаляр, связанный с матрицей, который даёт информацию о ее обратимости. Если определитель равен нулю, матрица необратима.

20. Средства NumPy для нахождения определителя

Для нахождения определителя матрицы используется функция `np.linalg.det()`:

np.linalg.det(A)

21. Обратная матрица

Обратная матрица — это матрица, которая при умножении на исходную дает единичную матрицу. Для нахождения обратной матрицы используется метод `np.linalg.inv()`.

22. Свойства обратной матрицы

- Обратная матрица для произведения: `(A * B)^(-1) = B^(-1) * A^(-1)`.
- Обратная матрица для транспонированной матрицы: `(A^T)^(-1) = (A^(-1))^T`.

23. Средства NumPy для нахождения обратной матрицы

Для нахождения обратной матрицы в NumPy используется:

np.linalg.inv(A)

24. Метод Крамера для решения систем линейных уравнений

Алгоритм метода Крамера заключается в вычислении определителей матрицы коэффициентов и матриц, полученных заменой столбцов на столбец свободных членов. В NumPy можно вычислить определители и решить систему с помощью этого метода.

```
import numpy as np
A = np.array([[2, 1], [1, 3]])
b = np.array([1, 2])
det_A = np.linalg.det(A)
x1 = np.linalg.det(np.column_stack((b, A[:, 1]))) / det_A
x2 = np.linalg.det(np.column_stack((A[:, 0], b))) / det_A
```

25. Матричный метод для решения систем линейных уравнений

Матричный метод решения системы уравнений Ax=b заключается в нахождении вектора x как x=A^−1b.

```
import numpy as np
A = np.array([[2, 1], [1, 3]])
b = np.array([1, 2])
x = np.linalg.inv(A).dot(b)
```

Вывод: в ходе практической работы мы исследовали базовые возможности библиотеки NumPy языка программирования Python.