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

Департамент цифровых, робототехнических
систем и электроники
Институт перспективной инженерии

ОТЧЕТ
ПО ЛАБАРАТОРНОЙ РАБОТЕ №2
дисциплины «Искусственный интеллект и машинное обучение»

Выполнил:

Стародубцев Дмитрий Андреевич 2 курс, группа ИВТ-б-о-23-2,09.03.01 «Информатика и вычислительная техника», очная форма обучения.

_____

(подпись)

Руководитель практики:

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.

Ссылка на репозиторий: https://github.com/bearncfu/OLD

Ход работы

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.

```python
import numpy as np

# Создал массив размером 3x3, содержащий числа от 1 до 9
array = np.arange(1, 10).reshape(3, 3)

# Умножил все элементы массива на 2
array = array * 2

# Заменил все элементы больше 10 на 0
array[array > 10] = 0

# Вывел итоговый массив
print(array)
```

Рисунок 1 – Код 1 задания

```
[[ 2  4  6]
 [ 8 10  0]
 [ 0  0  0]]
```

Рисунок 2 – Результат задания

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.

```python
import numpy as np

# Создал массив из 20 случайных целых чисел от 1 до 100
array = np.random.randint(1, 101, size=20)

# Вывел исходный массив
print("Исходный массив:")
print(array)

# Нашел элементы, которые делятся на 5 без остатка
array_5 = array[array % 5 == 0]

# Вывел элементы, которые делятся на 5
print("\nЭлементы, которые делятся на 5 без остатка:")
print(array_5)

# Заменил элементы, которые делятся на 5, на -1
array[array % 5 == 0] = -1

# Вывел обновленный массив
print("\nОбновленный массив:")
print(array)
```

Рисунок 3 – Код 2 задания

```
Исходный массив:
[35 99 93 76  6 19 94 15 22  9 14  4 34 32 43 58 33  3 48 18]

Элементы, которые делятся на 5 без остатка:
[35 15]

Обновленный массив:
[-1 99 93 76  6 19 94 -1 22  9 14  4 34 32 43 58 33  3 48 18]
```

Рисунок 4 – Результат 2 задания

Задание 3. Объединение и разбиение массивов.

Создайте два массива NumPy размером 1×5, заполненные случайными числами от 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.

```python
import numpy as np

# Создал два массива размером 1x5, заполненные случайными числами от 0 до 50
array1 = np.random.randint(0, 51, size=(1, 5))
array2 = np.random.randint(0, 51, size=(1, 5))

# Вывел исходные массивы
print("Исходный массив 1:")
print(array1)
print("\nИсходный массив 2:")
print(array2)

# Объединил массивы по строкам
two_array = np.vstack((array1, array2))

# Вывел объединенный массив
print("\nОбъединенный массив:")
print(two_array)

# Разделил объединенный массив на два массива
split_array1, split_array2 = np.split(two_array, 2, axis=0)

# Вывел разделенные массивы
print("\nРазделенный массив 1:")
print(split_array1)
print("\nРазделенный массив 2:")
print(split_array2)
```

Рисунок 5 – Код 3 задания

```
Исходный массив 1:
[[46 44  8 28 10]]

Исходный массив 2:
[[26  8 41 28 11]]

Объединенный массив:
[[46 44  8 28 10]
 [26  8 41 28 11]]

Разделенный массив 1:
[[46 44  8 28 10]]

Разделенный массив 2:
[[26  8 41 28 11]]
```

Рисунок 6 – Результат 3 задания

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.

```python
import numpy as np

# Создал массив из 50 чисел, равномерно распределенных от -10 до 10
array = np.linspace(-10, 10, 50)

# Вывел массив
print("Массив:")
print(array)

# Вычислил сумму всех элементов
sum_1 = np.sum(array)

# Вычислил сумму положительных элементов
sum_2 = np.sum(array[array > 0])

# Вычислил сумму отрицательных элементов
sum_3 = np.sum(array[array < 0])

# Вывел результаты
print("\nСумма всех элементов:", sum_1)
print("Сумма положительных элементов:", sum_2)
print("Сумма отрицательных элементов:", sum_3)
```

Рисунок 7 – Код 4 задания

```
Массив:
[-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: 7.105427357601002e-15
Сумма положительных элементов: 127.55102040816328
Сумма отрицательных элементов: -127.55102040816327
```

Рисунок 8 – Результат 4 задания

Задание 5. Работа с диагональными и единичными матрицами.

Создайте:

Единичную матрицу размером 4×4.

Диагональную матрицу размером 4×4 с диагональными элементами [5, 10, 15, 20] (не использовать циклы).

Найдите сумму всех элементов каждой из этих матриц и сравните результаты.

```python
import numpy as np

# Создал единичную матрицу размером 4x4
one_matrix = np.eye(4)

# Создал диагональную матрицу с диагональными элементами [5, 10, 15, 20]
diag_elements = [5, 10, 15, 20]
diag_matrix = np.diag(diag_elements)

# Вывел матрицы
print("Единичная матрица:")
print(one_matrix )
print("\nДиагональная матрица:")
print(diag_matrix )

# Вычислил сумму всех элементов каждой матрицы
sum_one = np.sum(one_matrix)
sum_diag = np.sum(diag_matrix)

# Вывел суммы
print("\nСумма всех элементов единичной матрицы:", sum_one)
print("Сумма всех элементов диагональной матрицы:", sum_diag)

# Сравнил результаты
if sum_one > sum_diag:
    print("\nСумма элементов единичной матрицы больше.")
elif sum_one < sum_diag:
    print("\nСумма элементов диагональной матрицы больше.")
else:
    print("\nСуммы элементов матриц равны.")
```

Рисунок 9 – Код 5 задания

```
Единичная матрица:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]

Диагональная матрица:
[[ 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

Сумма элементов диагональной матрицы больше.
```

Рисунок 10 - Результат 5 задания

Задание 6. Создание и базовые операции с матрицами.

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

Их сумму.

Их разность.

Их поэлементное произведение.

```python
import numpy as np

# Создал две квадратные матрицы размером 3x3, заполненные случайными целыми числами от 1 до 20
matrix1 = np.random.randint(1, 21, size=(3, 3))
matrix2 = np.random.randint(1, 21, size=(3, 3))

# Вывел исходные матрицы
print("Матрица 1:")
print(matrix1)
print("\nМатрица 2:")
print(matrix2)

# Вычислил сумму матриц
matrix_sum = matrix1 + matrix2

# Вычислил разность матриц
matrix_diff = matrix1 - matrix2

# Вычислил поэлементное произведение матриц
matrix_pr = matrix1 * matrix2

# Вывел результаты
print("\nСумма матриц:")
print(matrix_sum)
print("\nРазность матриц:")
print(matrix_diff)
print("\nПоэлементное произведение матриц:")
print(matrix_pr)
```

Рисунок 11 – Код 6 задания

```
Матрица 1:
[[ 5 13 16]
 [16 14  8]
 [19 18  4]]

Матрица 2:
[[ 7 20  5]
 [13 20  9]
 [18  3  1]]

Сумма матриц:
[[12 33 21]
 [29 34 17]
 [37 21  5]]

Разность матриц:
[[-2 -7 11]
 [ 3 -6 -1]
 [ 1 15  3]]

Поэлементное произведение матриц:
[[ 35 260  80]
 [208 280  72]
 [342  54   4]]
```

Рисунок 12 – Результат 6 задания

Задание 7. Умножение матриц.

Создайте две матрицы NumPy:

Первую размером 2×3, заполненную случайными числами от 1 до 10.

Вторую размером 3×2, заполненную случайными числами от 1 до 10.

Выполните матричное умножение (@ или np.dot) и выведите результат.

```python
import numpy as np

# Создал первую матрицу размером 2x3, заполненную случайными числами от 1 до 10
matrix1 = np.random.randint(1, 11, size=(2, 3))

# Создал вторую матрицу размером 3x2, заполненную случайными числами от 1 до 10
matrix2 = np.random.randint(1, 11, size=(3, 2))

# Вывел исходные матрицы
print("Матрица 1 (2x3):")
print(matrix1)
print("\nМатрица 2 (3x2):")
print(matrix2)

# Выполнил матричное умножение
result_matrix = np.dot(matrix1, matrix2)

# Вывел результат матричного умножения
print("\nРезультат матричного умножения (2x2):")
print(result_matrix)
```

Рисунок 13 – Код 7 задания

```
Матрица 1 (2x3):
[[9 8 3]
 [3 8 7]]

Матрица 2 (3x2):
[[ 7  8]
 [10  4]
 [ 4  3]]

Результат матричного умножения (2x2):
[[155 113]
 [129  77]]
```

Рисунок 14 – Результат 7 задания

Задание 8. Определитель и обратная матрица.

Создайте случайную квадратную матрицу 3×3. 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.

```python
import numpy as np

# Создал случайную квадратную матрицу 3x3
matrix = np.random.randint(1, 10, size=(3, 3))

# Вывел исходную матрицу
print("Исходная матрица:")
print(matrix)

# Вычислил определитель матрицы
det_matrix = np.linalg.det(matrix)

# Вывел определитель
print("\nОпределитель матрицы:")
print(det_matrix)

# Проверил, является ли матрица вырожденной (определитель равен 0)
if det_matrix == 0:
    print("\nМатрица вырождена, обратной матрицы не существует.")
else:
    # Вычислил обратную матрицу
    inverse_matrix = np.linalg.inv(matrix)
    print("\nОбратная матрица:")
    print(inverse_matrix)
```

Рисунок 15 – Код 8 задания

```
Исходная матрица:
[[8 2 8]
 [6 9 7]
 [9 4 9]]

Определитель матрицы:
-14.000000000000004

Обратная матрица:
[[-3.78571429 -1.          4.14285714]
 [-0.64285714  0.          0.57142857]
 [ 4.07142857  1.         -4.28571429]]
```

Рисунок 16 – Результат 8 задания

Задание 9. Транспонирование и след матрицы.

Создайте матрицу NumPy размером 4×4, 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.

```python
import numpy as np

# Создал матрицу размером 4x4, заполненную случайными целыми числами от 1 до 50
matrix = np.random.randint(1, 51, size=(4, 4))

# Вывел исходную матрицу
print("Исходная матрица:")
print(matrix)

# Вычислил транспонированную матрицу
trans_matrix = matrix.T

# Вывел транспонированную матрицу
print("\nТранспонированная матрица:")
print(trans_matrix)

# Вычислил след матрицы (сумму элементов на главной диагонали)
trace = np.trace(matrix)

# Выводим след матрицы
print("\nСлед матрицы (сумма элементов на главной диагонали):")
print(trace)
```

Рисунок 17 – Код 9 задания

```
Исходная матрица:
[[38  2 13 33]
 [40 13 32 49]
 [ 1 13 22 35]
 [12  6 26 16]]

Транспонированная матрица:
[[38 40  1 12]
 [ 2 13 13  6]
 [13 32 22 26]
 [33 49 35 16]]

След матрицы (сумма элементов на главной диагонали):
89
```

Рисунок 18 – Результат 9 задания

Задание 10. Системы линейных уравнений.

Решите систему линейных уравнений вида:

$$\begin{cases} 2x + 3y - z = 5 \\ 4x - y + 2z = 6 \\ -3x + 5y + 4z = -2 \end{cases}$$

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.

```python
import numpy as np

# Задал матрицу коэффициентов A
A = np.array([
    [2, 3, -1],
    [4, -1, 2],
    [-3, 5, 4]
])

# Задал вектор правой части B
B = np.array([5, 6, -2])

# Решил систему линейных уравнений
x = np.linalg.solve(A, B)

# Вывел результат
print("Решение системы:")
print("x =", x[0])
print("y =", x[1])
print("z =", x[2])
```

Рисунок 19 – Код 10 задания

```
Решение системы:
x = 1.6396396396396398
y = 0.5765765765765767
z = 0.009009009009008978
```

Рисунок 20 – Результат 10 задания

Задание 11. Вариант 14. Балансировка бюджета домашнего хозяйства.

Семья тратит свой месячный бюджет на три основные категории: питание, жилье и транспорт. На жилье уходит в два раза больше, чем на транспорт, а на питание — на 30% больше, чем на жилье. Если общий бюджет 150 000 рублей, сколько тратится на каждую категорию?

```python
from sympy import symbols, Eq, solve

# Определил переменные
transport = symbols('transport')
housing = symbols('housing')
food = symbols('food')

# Задал уравнения на основе условий задачи
# 1. На жилье уходит в два раза больше, чем на транспорт
eq1 = Eq(housing, 2 * transport)

# 2. На питание уходит на 30% больше, чем на жилье
eq2 = Eq(food, housing + 0.3 * housing)

# 3. Общий бюджет равен 150 000 рублей
eq3 = Eq(transport + housing + food, 150000)

# Решил систему уравнений
solution = solve((eq1, eq2, eq3), (transport, housing, food))

# Вывел результат
print("Расходы на транспорт:", round(solution[transport], 2), "руб.")
print("Расходы на жилье:", round(solution[housing], 2), "руб.")
print("Расходы на питание:", round(solution[food], 2), "руб.")
```

Рисунок 21 – Код 11 задания

```
Расходы на транспорт: 26785.71 руб.
Расходы на жилье: 53571.43 руб.
Расходы на питание: 69642.86 руб.
```

Рисунок 22 – Результат 11 задания

Контрольные вопросы:

1. Каково назначение библиотеки NumPy?

NumPy (Numerical Python) — это библиотека для работы с многомерными массивами и матрицами, а также для выполнения математических операций над ними. Основные функции:

- Эффективное хранение и обработка больших объемов данных.
- Выполнение математических операций (линейная алгебра, статистика, тригонометрия и т.д.).
- Интеграция с другими библиотеками, такими как SciPy, Pandas, Matplotlib.

2. Что такое массивы ndarray?

ndarray (N-dimensional array) — это многомерный массив в NumPy. Он представляет собой:

- Таблицу элементов одного типа (например, целые числа, числа с плавающей точкой).
- Быстрый доступ к данным благодаря оптимизированной реализации на C.
- Поддержка многомерных структур (векторы, матрицы, тензоры).

3. Как осуществляется доступ к частям многомерного массива?

Доступ к элементам массива осуществляется с помощью индексов. Для многомерных массивов используются несколько индексов, разделенных запятыми.

4. Как осуществляется расчет статистик по данным?

NumPy предоставляет множество функций для расчета статистик:

- Среднее значение: np.mean(arr)
- Медиана: np.median(arr)
- Стандартное отклонение: np.std(arr)
- Минимум и максимум: np.min(arr), np.max(arr)
- Сумма: np.sum(arr)

5. Как выполняется выборка данных из массивов ndarray?

Выборка данных может выполняться с помощью:

- Индексов: arr[0] (первый элемент).
- Срезов: arr[1:3] (элементы с индексами 1 и 2).
- Булевых масок: arr[arr > 2] (элементы больше 2).
- Индексации массивами: arr[[0, 2]] (элементы с индексами 0 и 2).

6. Основные виды матриц и векторов. Способы их создания в Python.

Виды матриц и векторов:

- Вектор: Одномерный массив.
- Матрица: Двумерный массив.
- Тензор: Многомерный массив (3D и выше).

Способы создания:

- Вектор: vector = np.array([1, 2, 3])
- Матрица: matrix = np.array([[1, 2], [3, 4]])
- Специальные матрицы:
  - Единичная матрица: np.eye(3)
  - Нулевая матрица: np.zeros((2, 2))
  - Матрица из единиц: np.ones((2, 2))

7. Как выполняется транспонирование матриц?

Транспонирование матрицы меняет строки и столбцы местами. В NumPy это делается с помощью: метода .T, функции np.transpose().

8. Свойства операции транспонирования матриц

1. $(A^T)^T = A$: Двойное транспонирование возвращает исходную матрицу.
2. $(A + B)^T = A^T + B^T$: Транспонирование суммы равно сумме транспонированных матриц.
3. $(kA)^T = kA^T$: Транспонирование произведения матрицы на скаляр равно произведению скаляра на транспонированную матрицу.
4. $(AB)^T = B^TA^T$: Транспонирование произведения матриц равно произведению транспонированных матриц в обратном порядке.

9. Средства в библиотеке NumPy для выполнения транспонирования матриц

В NumPy это делается с помощью: метода .T, функции np.transpose(), Функция np.swapaxes(): Меняет местами две оси массива.

10. Основные действия над матрицами.

Основные операции над матрицами включают:

- Умножение матрицы на число.
- Сложение и вычитание матриц.
- Умножение матриц.
- Транспонирование матриц.
- Вычисление определителя матрицы.

- Нахождение обратной матрицы.

11. Как осуществляется умножение матрицы на число?

Умножение матрицы на число выполняется поэлементно. Каждый элемент матрицы умножается на это число.

12. Свойства операции умножения матрицы на число

1. Ассоциативность: k·(l·A)=(k·l)·A$k·(l·A)=(k·l)·A$.
2. Дистрибутивность относительно сложения матриц: k·(A+B)=k·A+k·B$k·(A+B)=k·A+k·B$.
3. Дистрибутивность относительно сложения чисел: (k+l)·A=k·A+l·A$(k+l)·A=k·A+l·A$.

13. Как осуществляются операции сложения и вычитания матриц?

Сложение и вычитание матриц выполняются поэлементно. Матрицы должны иметь одинаковые размеры.

14. Свойства операций сложения и вычитания матриц.

1. Коммутативность сложения: A+B=B.
2. Ассоциативность сложения: (A+B)+C=A+(B+C).
3. Нейтральный элемент: A+0=A, где 0 — нулевая матрица.
4. Обратный элемент: A+(−A)= 0.

15. Средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц:

- Сложение: Используется оператор +.
- Вычитание: Используется оператор -.

16. Как осуществляется операция умножения матриц?

Умножение матриц выполняется по правилу "строка на столбец". Элемент Cij результирующей матрицы CC вычисляется как сумма произведений элементов ii-й строки первой матрицы на элементы jj-го столбца второй матрицы.

17. Свойства операции умножения матриц

1. Ассоциативность: (A·B)·C=A·(B·C) .
2. Дистрибутивность: A·(B+C)=A·B+A·C.

3. Не коммутативность: В общем случае $A \cdot B \neq B \cdot A$ $A \cdot B = B \cdot A$.

18. Средства в библиотеке NumPy для выполнения операции умножения матриц.

Функция np.dot(), оператор @.

19. Что такое определитель матрицы? Каковы свойства определителя матрицы?

Определитель матрицы — это скалярная величина, которая характеризует свойства квадратной матрицы. Определитель используется для решения систем линейных уравнений, нахождения обратной матрицы и анализа линейной независимости векторов.

Свойства определителя:

1. Определитель единичной матрицы равен 1.
2. Определитель транспонированной матрицы равен определителю исходной матрицы.
3. Если две строки (или столбца) матрицы одинаковы, определитель равен 0.
4. Определитель произведения матриц равен произведению их определителей: $\det(A \cdot B) = \det(A) \cdot \det(B)$.

20. Средства в библиотеке NumPy для нахождения значения определителя матрицы.

Для вычисления определителя используется функция np.linalg.det().

21. Что такое обратная матрица? Какой алгоритм нахождения обратной матрицы?

Обратная матрица — это такая матрица A^{-1}, которая при умножении на исходную матрицу A дает единичную матрицу E:

$A \cdot A^{-1} = A^{-1} \cdot A = E$ $A \cdot A^{-1} = A^{-1} \cdot A = E$

Обратная матрица существует только для квадратных матриц, определитель которых не равен нулю (т.е. для невырожденных матриц).

Алгоритм нахождения обратной матрицы:

1. Проверить, что матрица квадратная и её определитель не равен нулю.

2. Найти матрицу алгебраических дополнений (союзную матрицу).

3. Транспонировать союзную матрицу (получить присоединённую матрицу).

4. Разделить каждый элемент присоединённой матрицы на определитель исходной матрицы.

22. Каковы свойства обратной матрицы?

1. Уникальность: Если обратная матрица существует, то она единственная.

2. Обратная к обратной: $(A^{-1})^{-1}=A(A^{-1})^{-1}=A$.

3. Обратная к произведению: $(AB)^{-1}=B^{-1}A^{-1}(AB)^{-1}=B^{-1}A^{-1}$.

4. Обратная к транспонированной: $(AT)^{-1}=(A^{-1})^{T}$.

5. Определитель обратной матрицы: $\det(A^{-1})= 1/\det(A)$.

23. Какие имеются средства в библиотеке NumPy для нахождения обратной матрицы?

В библиотеке NumPy для нахождения обратной матрицы используется функция numpy.linalg.inv.

24. Метод Крамера для решения систем линейных уравнений

Метод Крамера — это метод решения систем линейных уравнений с использованием определителей. Он применим только для систем, где число уравнений равно числу неизвестных, и определитель матрицы системы не равен нулю.

25. Матричный метод для решения систем линейных уравнений

Матричный метод заключается в решении системы линейных уравнений с использованием обратной матрицы. Если система задана в виде $A\cdot X=B$, то решение можно найти как $X=A^{-1}\cdot B$.

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

работы были выполнены следующие задания: создание и изменение массивов, работа с булевыми масками, объединение и разбиение массивов, генерация и работа с линейными последовательностями, работа с диагональными и единичными матрицами, создание и базовые операции с матрицами, умножение матриц, определитель и обратная матрица, транспонирование и след матрицы, системы линейных уравнений.