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 ЛАБОРАТОРНОЙ РАБОТЕ №2
дисциплины
«Искусственный интеллект и машинное обучение»
Вариант № 4

Выполнил:
Левашев Тимур Рашидович
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 В.И

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема работы: “Основы работы с библиотекой NumPy”.

Цель работы: исследовать базовые возможности библиотеки NumPy языка программирования Python.

Порядок выполнения работы:

Ссылка на git репозиторий: https://github.com/mazy99/2_laba_ml

1. Выполнил 1 задание из методического материала:

1. Создание и изменение массивов

Описание задания:

- 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.

[56]:
```
import numpy as np
array = np.random.randint(1, 10, size = (3, 3))
print(f"Матрица 3 на 3 с рандомными числами от 1 до 9:\n {array}")
array = 2 * array
print(f"Исходная матрица,все элементы которой умножены на 2:\n {array}")
array[array > 10] = 0
print(f"Исходная матрица,все элементы которой умножены на 2 и все элементы большие 10 заменены на 0:\n {array}")
```

Рисунок 1 – Программа к заданию № 1

```
Матрица 3 на 3 с рандомными числами от 1 до 9:
 [[7 6 5]
 [7 2 4]
 [8 1 9]]
Исходная матрица,все элементы которой умножены на 2:
 [[14 12 10]
 [14  4  8]
 [16  2 18]]
Исходная матрица,все элементы которой умножены на 2 и все элементы большие 10 заменены на 0:
 [[ 0  0 10]
 [ 0  4  8]
 [ 0  2  0]]
```

Рисунок 2 – Результат выполнения программы для задания 1

2. Выполнил задание 2 из методического материала:

2.Работа с булевыми масками

Описание задания:

- 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

```
[61]: array = np.random.randint(1, 101, size=(1,20))
      print(f"Массив из 20 случайных чисел от 1 до 100:\n {array}")
      print(f"Массив, в котором все элементы кратны 5:\n {array[array%5==0]}")
      array[array%5==0]=-1
      print(f"Массив, в котором все элементы кратные 5 заменены на -1:\n {array}")
```

Рисунок 3 – Программа к заданию № 2

```
Массив из 20 случайных чисел от 1 до 100:
 [[75 78 87 17 43 36 88 24 43 25 59 65 38 50 68 79 75 11 44 57]]
Массив, в котором все элементы кратны 5:
 [75 25 65 50 75]
Массив, в котором все элементы кратные 5 заменены на -1:
 [[-1 78 87 17 43 36 88 24 43 -1 59 -1 38 -1 68 79 -1 11 44 57]]
```

Рисунок 4 – Результат выполнения программы для задания 2

3. Выполнил задание 3 из методического материала:

3. Объединение и разбиение массивов

Описание задания

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

Элементы задания: ¶

- **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.

Выведите все промежуточные и итоговые результаты.

```
[69]: array_1 = np.random.randint(0, 51, size = (1, 5))
      print(f"Первый массив: {array_1}")
      array_2 = np.random.randint(0, 51, size = (1, 5))
      print(f"Второй массив: {array_2}")
      combi_array = np.vstack((array_1,array_2))
      print(f"Объединенный массив:\n {combi_array}")
      split_array = np.vsplit(combi_array,2)
      print(f"Разделенные массивы:\n {split_array}")
```

Рисунок 5 – Программа к заданию № 3

```
Первый массив: [[ 1 31  5 13 46]]
Второй массив: [[ 3  7 20  4 20]]
Объединенный массив:
 [[ 1 31  5 13 46]
 [ 3  7 20  4 20]]
Разделенные массивы:
 [array([[ 1, 31,  5, 13, 46]]), array([[ 3,  7, 20,  4, 20]])]
```

Рисунок 4 – Результат выполнения программы для задания 3

4. Выполнил задание 4 из методического материала:

4.Генерация и работа с линейными последовательностями

Описание задания:

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

```
[77]: array = np.linspace(-10,10,50)
print(f"Исходный массив с равномерно распределенными числами от -10 до 10:\n {array}")
sum_array = np.sum(array)
print(f"Сумма всех элементов массива: {sum_array}")
positive_elements = array[array > 0]
sum_positive = np.sum(positive_elements)
print(f"Сумма положительных элементов массива: {sum_positive}")
negative_elements = array[array < 0]
sum_negative = np.sum(negative_elements)
print(f"Сумма отрицательных элементов массива: {sum_negative}")
```

Рисунок 5 – Программа к заданию № 4

```
Исходный массив с равномерно распределенными числами 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 положительных элементов массива: 127.55102040816328
Сумма отрицательных элементов массива: -127.55102040816327
```

Рисунок 6 – Результат выполнения программы для задания 4

5. Выполнил задание 5 из методического материала:

5.Работа с диагональными и единичными матрицами

Описание задания:

Создайте:

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

```
[80]: eye_matrix = np.eye(4)
print(f"Единичная матрица 4 на 4:\n {eye_matrix}")
nums_diag = np.array([5,10,15,20])
diag_matrix = np.diag(nums_diag)
print(f"Диагональная матрица 4 на 4:\n {diag_matrix}")
```

Рисунок 7 – Программа к заданию № 5

```
Единичная матрица 4 на 4:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
Диагональная матрица 4 на 4:
 [[ 5  0  0  0]
 [ 0 10  0  0]
 [ 0  0 15  0]
 [ 0  0  0 20]]
```

Рисунок 8 – Результат выполнения программы к заданию 5

6. Выполнил задание 6 из методического материала:

6.Создание и базовые операции с матрицами

Описание задания:

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

```
[90]: matrix_1 = np.random.randint(1, 21, size = (3, 3))
print(f"Первая матрица:\n {matrix_1}")
matrix_2 = np.random.randint(1, 21, size = (3, 3))
print(f"Вторая матрица:\n {matrix_2}")
sum_matrix = np.add(matrix_1, matrix_2)
print(f"Сумма матриц:\n {sum_matrix}")
diff_matrix_1 = np.subtract(matrix_1, matrix_2)
print(f"Разность между матрицам:\n {diff_matrix_1}")
multi_matrix = np.multiply(matrix_1,matrix_2)
print(f"Поэлементное произведенеие матриц:\n {multi_matrix}")
```

Would you like to c

Рисунок 9 – Программа к заданию № 6

```
Первая матрица:
 [[ 1  1  9]
 [12 15 20]
 [13 19 17]]
Вторая матрица:
 [[ 9 16 15]
 [ 9 15  4]
 [11 14 17]]
Сумма матриц:
 [[10 17 24]
 [21 30 24]
 [24 33 34]]
Разность между матрицам:
 [[ -8 -15  -6]
 [  3   0  16]
 [  2   5   0]]
Поэлементное произведенеие матриц:
 [[  9  16 135]
 [108 225  80]
 [143 266 289]]
```

Рисунок 10 – Результат выполнения программы к заданию 6

7. Выполнил задание 7 из методического материала:

7.Умножение матриц

Описание задания:

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.

[93]:
```
matrix_1 = np.random.randint(1, 11, size = (2, 3))
print(f"Первая матрица:\n {matrix_1}")
matrix_2 = np.random.randint(1, 10, size = (3, 2))
print(f"Вторая матрица:\n {matrix_2}")
multi_matrix_1 = np.dot(matrix_1,matrix_2)
print(f"Произведение матриц первой и второй:\n {multi_matrix_1}")
multi_matrix_2 = np.dot(matrix_2,matrix_1)
print(f"Произведение матриц второй и первой:\n {multi_matrix_2}")
```

Рисунок 11 – Программа к заданию № 7

```
Первая матрица:
 [[5 7 2]
 [2 8 1]]
Вторая матрица:
 [[8 3]
 [9 3]
 [8 9]]
Произведение матриц первой и второй:
 [[119  54]
 [ 96  39]]
Произведение матриц второй и первой:
 [[ 46  80  19]
 [ 51  87  21]
 [ 58 128  25]]
```

Рисунок 12 – Результат выполнения программы к заданию 7

8. Выполнил задание 8 из методического материала:

8.Опеределитель и обратная матрица

Описание задания:

Создайте случайную квадратную матрицу 3×3. Найдите и выведите:

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

[101]:
```python
matrix = np.random.randint(-10, 11, size = (3, 3))
print(f"Случайная матрица 3 на 3:\n {matrix}")
det_mat = np.linalg.det(matrix)
print(f"Определитель матрицы:{det_mat}")
if det_mat != 0:
    inv_mat = np.linalg.inv(matrix)
    print(f"Обратная матрица:\n {inv_mat}")
else:
    print(f"Детерминат матрицы равен {det_mat}, а значит матрица не выраждена и не имеет обратной")
```

Рисунок 13 – Программа к заданию № 8

```
Случайная матрица 3 на 3:
 [[ 9 -5  2]
 [-8  5  0]
 [-6  6  3]]
Определитель матрицы:-21.000000000000001
Обратная матрица:
 [[-0.71428571 -1.28571429  0.47619048]
 [-1.14285714 -1.85714286  0.76190476]
 [ 0.85714286  1.14285714 -0.23809524]]
```

Рисунок 14 – Результат выполнения программы к заданию 8

9. Выполнил задание 9 из методического материала:

9.Транспанирование и след матрицы

Описание задания:

Создайте матрицу NumPy размером 4×4, содержащую случайные целые числа от1 до 50.

Выведите:

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

```
107]: matrix = np.random.randint(1, 51, size = (4, 4))
print(f"Случайная матрица 4 на 4:\n {matrix}")
t_mat = np.transpose(matrix)
print(f"Транспонированная матрица:\n {t_mat}")
trace = np.trace(matrix)
print(f"След матрицы: {trace}")
```

Рисунок 15 – Программа к заданию № 9

```
Случайная матрица 4 на 4:
 [[14 37 21  2]
 [43 22  8  7]
 [40 33 26  2]
 [ 8 25 10 22]]
Транспонированная матрица:
 [[14 43 40  8]
 [37 22 33 25]
 [21  8 26 10]
 [ 2  7  2 22]]
След матрицы: 84
```

Рисунок 16 – Результат выполнения программы к заданию 9

10. Выполнил задание 10 из методического материала:

10.Системы линейных уравнений ¶

Описание задания:

Решите систему линейных уравнений вида:

$$2x + 3y - z = 5$$
$$4x - y + 2z = 6$$
$$-3x + 5y + 4z = -2$$

```
12]: A = np.array([[2,3,-1],[4,-1,2],[-3,5,4]])
print(f"Матрица коэффициентов:\n {A}")
B = np.array([5,6,-2]).reshape(-1,1)
print(f"Вектор правой части:\n {B}")
result = np.linalg.solve(A,B)
print(f"Решение системы:\n {result}")
```

Рисунок 17 – Программа к заданию № 10

```
Матрица коэффициентов:
 [[ 2  3 -1]
 [ 4 -1  2]
 [-3  5  4]]
Вектор правой части:
 [[ 5]
 [ 6]
 [-2]]
Решение системы:
 [[1.63963964]
 [0.57657658]
 [0.00900901]]
```

Рисунок 18 – Результат выполнения программы к заданию 10

11. Выполнил индивидуальное задание по вариантам из методического материала:

Описание задания:

Решите индивидуальное задание согласно варианта. Каждое задание предусматривает построение системы линейных уравнений. Решите полученную систему уравнений с использованием библиотеки NumPy. Для решения системы используйте метод Крамера и матричный метод. Сравните полученные результаты, с результатами, полученными с помощью np.linalg.solve .

4 задание:

Рацион питания. В диете необходимо включить три продукта: мясо, рыбу и овощи. Они содержат белки, жиры и углеводы в разном количестве. В 100 г мяса содержится 20 г белков, 10 г жиров и 5 г углеводов, в 100 г рыбы — 15 г белков, 5 г жиров и 10 г углеводов, в 100 г овощей — 5 г белков, 2 г жиров и 20 г углеводов. Суточная норма: 100 г белков, 50 г жиров и 150 г углеводов. Сколько граммов каждого продукта нужно съесть?

Исходя из данных получим следующую систему линейных уравнений:

$$20x + 15y + 5z = 100$$
$$10x + 5y + 2z = 50$$
$$5x + 10y + 20z = 150$$

2]:
```
import numpy as np
A = np.array([[20,15,5],[10,5,2],[5,10,20]])
print(f"Матрица коэффициентов:\n {A}")
B = np.array([100,50,150]).reshape(-1,1)
print(f"Вектор правой части:\n {B}")
result = np.linalg.solve(A,B)
print(f"Решение системы:\n {result}")
```

Рисунок 19 – Программа к индивидуальному заданию

```
Матрица коэффициентов:
 [[20 15  5]
 [10  5  2]
 [ 5 10 20]]
Вектор правой части:
 [[100]
 [ 50]
 [150]]
Решение системы:
 [[ 4.28571429]
 [-1.42857143]
 [ 7.14285714]]
```

Рисунок 20 – Результат выполнения программы к индивидуальному заданию

Ответы на 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: исследовал базовые возможности библиотеки NumPy языка программирования Python.