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 9

Выполнил:
Неутолимов Дмитрий Сергеевич
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.

Ссылка на репозиторий: https://github.com/SiXTeeN100/AIaML

Порядок выполнения работы:

Задание 1. Создание и изменение массивов

Создал массив NumPy размером 3×3, содержащий числа от 1 до 9. Умножил все элементы массива на 2. Заменил все элементы больше 10 на 0. Вывел итоговый массив.

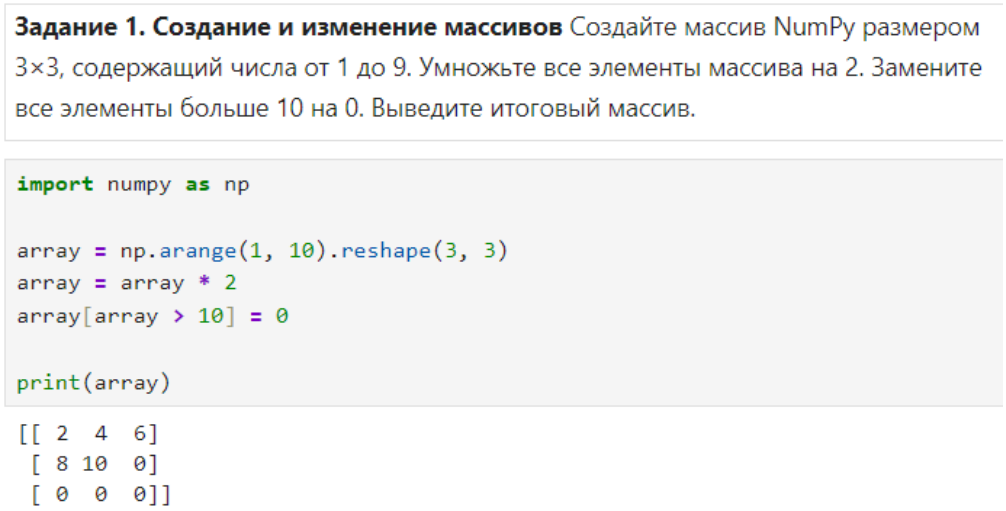

Рисунок 1 – Код задания 1

Задание 2. Работа с булевыми масками

Создал массив NumPy из 20 случайных целых чисел от 1 до 100. Нашёл и вывел все элементы, которые делятся на 5 без остатка. Заменил эти элементы на -1. Вывел обновленный массив.

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. Замените эти элементы на -1. Выведите обновленный массив.

```
import numpy as np

array = np.random.randint(1, 101, 20)
delOnFive = array[array % 5 == 0]

print("Элементы, делящиеся на 5:", delOnFive)

array[array % 5 == 0] = -1

print("Обновленный массив:", array)
```

```
Элементы, делящиеся на 5: [ 45  60  85  55 100]
Обновленный массив: [-1 11 23 11 88 68 73 94 -1 84 84 -1 54 26 28 43 13 -1 -1 1
7]
```

Рисунок 2 – Код задания 2

Задание 3. Объединение и разбиение массивов

Создал два массива NumPy размером 1×5, заполненные случайными числами от 0 до 50. Объединил эти массивы в один двумерный массив (по строкам). Разделил полученный массив на два массива, каждый из которых содержит 5 элементов. Вывел все промежуточные и итоговые результаты.

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.

```
import numpy as np

array1 = np.random.randint(0, 51, (1, 5))
array2 = np.random.randint(0, 51, (1, 5))

print("Массив 1:", array1)
print("Массив 2:", array2)

# Объединение массивов по строкам
combinedArray = np.vstack((array1, array2))

print("Объединенный массив:", combinedArray)

# Разделение массивов на два массива
splitArrays = np.split(combinedArray, 2)

print("Разделенный массив 1:", splitArrays[0])
print("Разделенный массив 2:", splitArrays[1])
```

```
Массив 1: [[47 26 17  3 15]]
Массив 2: [[19 21 26 16  1]]
Объединенный массив: [[47 26 17  3 15]
 [19 21 26 16  1]]
Разделенный массив 1: [[47 26 17  3 15]]
Разделенный массив 2: [[19 21 26 16  1]]
```

Рисунок 3 – Код задания 3

Задание 4. Генерация и работа с линейными последовательностями

Создал массив из 50 чисел, равномерно распределенных от -10 до 10. Вычислил сумму всех элементов массива. Вычислил сумму положительных элементов. Вычислил сумму отрицательных элементов. Вывел результаты.

Задание 4. Генерация и работа с линейными последовательностями Создайте массив из 50 чисел, равномерно распределенных от -10 до 10. Вычислите сумму всех элементов массива. Вычислите сумму положительных элементов. Вычислите сумму отрицательных элементов. Выведите результаты.

```
import numpy as np

array = np.linspace(-10, 10, 50)

Sum = np.sum(array)
positiveSum = np.sum(array[array > 0])
negativeSum = np.sum(array[array < 0])

print("Сумма всех элементов:", Sum)
print("Сумма положительных элементов:", positiveSum)
print("Сумма отрицательных элементов:", negativeSum)
```

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

Рисунок 4 – Код задания 4

Задание 5. Работа с диагональными и единичными матрицами

Создал единичную матрицу размером 4×4. Создал диагональную матрицу размером 4×4 с диагональными элементами [5, 10, 15, 20] (без использования циклов). Нашёл сумму всех элементов каждой из этих матриц. Сравнил результаты.

```python
import numpy as np

# Единичная матрица размером 4×4
singleMatrix = np.eye(4)

# Диагональная матрица с диагональными элементами [5, 10, 15, 20]
diagonalMatrix = np.diag([5, 10, 15, 20])

sumOfSingle = np.sum(singleMatrix)
sumOfDiagonal = np.sum(diagonalMatrix)

print("Единичная матрица:\n", singleMatrix)
print("Сумма элементов единичной матрицы:", sumOfSingle)

print("Диагональная матрица:\n", diagonalMatrix)
print("Сумма элементов диагональной матрицы:", sumOfDiagonal)

if sumOfSingle > sumOfDiagonal:
    print("Сумма элементов единичной матрицы больше.")
elif sumOfSingle < sumOfDiagonal:
    print("Сумма элементов диагональной матрицы больше.")
else:
    print("Суммы элементов матриц равны.")
```

```
Единичная матрица:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
Сумма элементов единичной матрицы: 4.0
Диагональная матрица:
 [[ 5  0  0  0]
 [ 0 10  0  0]
 [ 0  0 15  0]
 [ 0  0  0 20]]
Сумма элементов диагональной матрицы: 50
Сумма элементов диагональной матрицы больше.
```

Рисунок 5 – Код задания 5

Задание 6. Создание и базовые операции с матрицами

Создал две квадратные матрицы NumPy размером 3×3, заполненные случайными целыми числами от 1 до 20. Вычислил и вывел их сумму, разность, поэлементное произведение.

```python
import numpy as np

matrix1 = np.random.randint(1, 21, (3, 3))
matrix2 = np.random.randint(1, 21, (3, 3))

print("Первая матрица:\n", matrix1)
print("Вторая матрица:\n", matrix2)

sumMatrix = matrix1 + matrix2
print("Сумма матриц:\n", sumMatrix)

differenceOfMatrix = matrix1 - matrix2
print("Разность матриц:\n", differenceOfMatrix)

# Поэлементное произведение матриц
elementwiseProductOfMatrix = matrix1 * matrix2
print("Поэлементное произведение матриц:\n", elementwiseProductOfMatrix)
```

```
Первая матрица:
 [[ 1  3 14]
 [ 1 12  4]
 [16  5 11]]
Вторая матрица:
 [[14 13 19]
 [10  8 17]
 [19  1  8]]
Сумма матриц:
 [[15 16 33]
 [11 20 21]
 [35  6 19]]
Разность матриц:
 [[-13 -10  -5]
 [ -9   4 -13]
 [ -3   4   3]]
Поэлементное произведение матриц:
 [[ 14  39 266]
 [ 10  96  68]
 [304   5  88]]
```

Рисунок 6 – Код задания 6

Задание 7. Умножение матриц

Создал 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. Выполнил матричное умножение с использованием оператора @ или функции np.dot. Вывел результат.

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 с использованием оператора @ или функции np.dot. Выведите результат.

```
import numpy as np

matrix1 = np.random.randint(1, 11, (2, 3))
matrix2 = np.random.randint(1, 11, (3, 2))

print("Матрица 1 (2×3):\n", matrix1)
print("Матрица 2 (3×2):\n", matrix2)

resultMatrix = np.dot(matrix1, matrix2)

# Выводим результат
print("Результат матричного умножения:\n", resultMatrix)
```

```
Матрица 1 (2×3):
 [[ 4  4 10]
 [ 1  4  3]]
Матрица 2 (3×2):
 [[8 3]
 [6 1]
 [1 3]]
Результат матричного умножения:
 [[66 46]
 [35 16]]
```

Рисунок 7 – Код задания 7

Задание 8. Определитель и обратная матрица

Создал случайную квадратную матрицу 3×3. Нашёл и вывел определитель этой матрицы, обратную матрицу (если она существует, иначе вывожу сообщение, что матрица вырождена). Использовал функции np.linalg.det и np.linalg.inv.

```
import numpy as np

matrix = np.random.randint(1, 10, (3, 3))

print("Исходная матрица:\n", matrix)

det = np.linalg.det(matrix)
print("Определитель матрицы:", det)

# Проверка на то, существует ли обратная матрица
if det != 0:
    # Нахождение обратной матрицы
    inverseMatrix = np.linalg.inv(matrix)
    print("Обратная матрица:\n", inverseMatrix)
else:
    print("Матрица вырождена, обратной матрицы не существует.")
```

```
Исходная матрица:
 [[4 9 2]
 [8 8 7]
 [3 6 4]]
Определитель матрицы: -90.99999999999997
Обратная матрица:
 [[ 0.10989011  0.26373626 -0.51648352]
 [ 0.12087912 -0.10989011  0.13186813]
 [-0.26373626 -0.03296703  0.43956044]]
```

Рисунок 8 – Код задания 8

Задание 9. Транспонирование и след матрицы

Создал матрицу NumPy размером 4×4, содержащую случайные целые числа от 1 до 50. Вывел исходную матрицу, транспонированную матрицу, след матрицы (сумму элементов на главной диагонали). Использовал функцию np.trace для нахождения следа.

```
import numpy as np

matrix = np.random.randint(1, 51, (4, 4))
print("Исходная матрица:\n", matrix)

transposedMatrix = matrix.T
print("Транспонированная матрица:\n", transposedMatrix)

traceOfMatrix = np.trace(matrix)
print("След матрицы:", traceOfMatrix)
```

```
Исходная матрица:
 [[13 38  7 25]
 [22 14  7 30]
 [36 29 33 41]
 [34 39  9 10]]
Транспонированная матрица:
 [[13 22 36 34]
 [38 14 29 39]
 [ 7  7 33  9]
 [25 30 41 10]]
След матрицы: 70
```

Рисунок 9 – Код задания 9

Задание 10. Системы линейных уравнений

Решил систему линейных уравнений вида:

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

Использовал матричное представление Ax = B, где: A — матрица коэффициентов, x — вектор неизвестных, B — вектор правой части. Решил систему с помощью функции np.linalg.solve и вывел результат.

```python
import numpy as np

A = np.array([[2, 3, -1],
              [4, -1, 2],
              [-3, 5, 4]])
B = np.array([5, 6, -2])
x = np.linalg.solve(A, B)

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

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

Рисунок 10 – Код задания 10

Задание 11. Индивидуальное задание. Вариант 9.

Оптимальный выбор топлива. Автомобиль использует три вида топлива в разных пропорциях. Бензин даёт 30 МДж энергии на литр, дизель — 35 МДж, газ — 25 МДж. Для поездки требуется 100 МДж энергии. Водитель хочет использовать бензина в два раза больше, чем газа, а количество дизеля равно количеству газа. Сколько литров каждого топлива нужно залить?

```python
# Общее уравнение: 60x + 35x + 25x = 100
# Упрощаем: 120x = 100
x = 100 / 120

# Количество каждого топлива
gas = x
petrol = 2 * x
diesel = x

# Выводим результаты
print(f"Количество газа: {gas:.3f} литра")
print(f"Количество бензина: {petrol:.3f} литра")
print(f"Количество дизеля: {diesel:.3f} литра")
```

```
Количество газа: 0.833 литра
Количество бензина: 1.667 литра
Количество дизеля: 0.833 литра
```

Рисунок 11 – Код индивидуального задания

Ответы на контрольные вопросы:

1. Каково назначение библиотеки NumPy?

NumPy (Numerical Python) — это библиотека для языка Python, предназначенная для работы с многомерными массивами и матрицами. Она предоставляет функции для выполнения математических операций, линейной алгебры, статистики и других вычислений.

2. Что такое массивы ndarray?

ndarray (n-dimensional array) — это основной объект в NumPy, представляющий собой многомерный массив элементов одного типа. Он позволяет эффективно хранить и обрабатывать большие объемы данных.

3. Как осуществляется доступ к частям многомерного массива?

Доступ к элементам или частям массива осуществляется с помощью индексации и срезов. Например, array[0, 1] позволяет получить элемент из первой строки и второго столбца.

4. Как осуществляется расчет статистик по данным?

NumPy предоставляет функции для расчета статистик, такие как np.mean(), np.median(), np.std() (стандартное отклонение), np.sum() и другие.

5. Как выполняется выборка данных из массивов ndarray?

Выборка данных может быть выполнена с помощью индексации, срезов или булевых масок. Например, array[array > 5] вернет все элементы массива, которые больше 5.

6. Приведите основные виды матриц и векторов. Опишите способы их создания в языке Python.

Вектор: одномерный массив.

Матрица: двумерный массив.

Способы создания:

Вектор: np.array([1, 2, 3])

Матрица: np.array([[1, 2], [3, 4]]) или np.zeros((2, 2)) для нулевой матрицы.

7. Как выполняется транспонирование матриц?

Транспонирование выполняется с помощью метода. T или функции np.transpose(). Например, matrix.T вернет транспонированную матрицу.

8. Приведите свойства операции транспонирования матриц.

$$(A + B)^T = A^T + B^T$$

$$(A^T)^T = A$$

$(kA)^T = kA^T$, где k – скаляр

9. Какие имеются средства в библиотеке NumPy для выполнения транспонирования матриц?

В NumPy для транспонирования можно использовать атрибут .T или функцию np.transpose().

10. Какие существуют основные действия над матрицами?

Сложение и вычитание.

Умножение на скаляр.

Умножение матриц.

Транспонирование.

Нахождение обратной матрицы.

11. Как осуществляется умножение матрицы на число?

Умножение выполняется поэлементно. Например, matrix * 2 умножит каждый элемент матрицы на 2.

12. Какие свойства операции умножения матрицы на число?

Ассоциативность: $k(lA) = (kl)A$

Дистрибутивность: $k(A + B) = kA + kB$

13. Как осуществляется операции сложения и вычитания матриц?

Операции выполняются поэлементно. Например, A + B сложит соответствующие элементы матриц A и B.

14. Каковы свойства операций сложения и вычитания матриц?

Коммутативность сложения: $A + B = B + A$

Ассоциативность сложения: $(A + B) + C = A + (B + C)$

Нейтральный элемент: $A + 0 = A$, где 0 — нулевая матрица.

15. Какие имеются средства в библиотеке NumPy для выполнения операций сложения и вычитания матриц?

В NumPy сложение и вычитание матриц выполняются с помощью операторов + и -. Например, A + B и A - B выполняют поэлементное сложение и вычитание соответственно.

16. Как осуществляется операция умножения матриц?

Умножение матриц выполняется с использованием функции np.dot() или оператора @. Например, np.dot(A, B) или A @ B выполнят умножение матриц A и B.

17. Каковы свойства операции умножения матриц?

Ассоциативность: $(AB)C = A(BC)$

Дистрибутивность: $A(B + C) = AB + AC$

18. Какие имеются средства в библиотеке NumPy для выполнения операции умножения матриц?

В NumPy для умножения матриц используются функции np.dot(), np.matmul() или оператор @.

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

Определитель матрицы — это скалярное значение, которое может быть вычислено только для квадратных матриц. Оно используется для определения обратимости матрицы и других свойств.

Свойства определителя:

Определитель единичной матрицы равен 1.

Если матрица имеет нулевой определитель, она вырожденная (не обратимая).

Определитель произведения матриц равен произведению их определителей: $\det(AB) = \det(A) * \det(B)$

20. Какие имеются средства в библиотеке NumPy для нахождения значения определителя матрицы?

В NumPy определитель матрицы можно найти с помощью функции np.linalg.det(). Например, np.linalg.det(A) вернет определитель матрицы A.

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

Обратная матрица A^{-1} — это такая матрица, что $A * A^{-1} = I$, где I — единичная матрица. Обратная матрица существует только для невырожденных матриц (с ненулевым определителем).

Алгоритм нахождения: Обратная матрица может быть найдена с помощью метода Гаусса-Жордана или через алгебраические дополнения.

22. Каковы свойства обратной матрицы?

$$(A^{-1})^{-1} = A$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

$(kA)^{-1} = \frac{1}{k}A^{-1}$, где k – скаляр

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

В NumPy обратная матрица может быть найдена с помощью функции np.linalg.inv(). Например, np.linalg.inv(A) вернет обратную матрицу для A.

24. Самостоятельно изучите метод Крамера для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений методом Крамера средствами библиотеки NumPy.

Метод Крамера используется для решения систем линейных уравнений, где число уравнений равно числу неизвестных. Основная идея метода заключается в использовании определителей матриц.

Алгоритм решения системы линейных уравнений методом Крамера:

1) Записать систему уравнений в матричной форме AX=B, где:
   - A — матрица коэффициентов,
   - X — вектор неизвестных,
   - B — вектор свободных членов.

2) Вычислить определитель матрицы A (det(A)). Если det(A)=0, система либо не имеет решений, либо имеет бесконечно много решений.

3) Для каждого неизвестного xi создать матрицу Ai, заменив i-й столбец матрицы A на вектор B.

4) Вычислить определитель каждой матрицы Ai (det(Ai)).

5) Найти каждое неизвестное по формуле:

$$x_i = \frac{\det(A_i)}{\det(\mathrm{A})}$$

25. Самостоятельно изучите матричный метод для решения систем линейных уравнений. Приведите алгоритм решения системы линейных уравнений матричным методом средствами библиотеки NumPy.

Матричный метод (метод обратной матрицы) используется для решения систем линейных уравнений, где число уравнений равно числу неизвестных. Основная идея метода заключается в нахождении обратной матрицы.

Алгоритм решения системы линейных уравнений матричным методом:

1) Записать систему уравнений в матричной форме AX=B, где: A — матрица коэффициентов, X — вектор неизвестных, B — вектор свободных членов.

2) Найти обратную матрицу A^{-1}.

3) Умножить обратную матрицу на вектор B:

$$X = A^{-1} * B$$

Вывод: в ходе лабораторной работы были исследованы базовые возможности библиотеки NumPy языка программирования Python. Сделаны задания, связанные с библиотекой NumPy. Сделано индивидуальное задание.