МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО «СЕВЕРО - КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ
КАФЕДРА ИНФОКОММУНИКАЦИЙ

«Основы работы с библиотекой NumPy»

Отчет

по лабораторной работе №2

дисциплины

«Теория распознаваныя образов»

Выполнил:
Гълбачева Доротея Андреева
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения
_____
(подпись) Проверил:
_____
(подпись)

Отчет защищен с оценкой_____ Дата защиты_____

Ставрополь, 2023 г.

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

Выполнения лабораторной работы:

1. Проработка примеров из лабораторной работы:

```
In [1]: import numpy as np

In [2]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 2 5 7')
        print(m)

        [[1 2 3 4]
         [5 6 7 8]
         [9 2 5 7]]
```

Рисунок 2.1 – Импортируем библиотеку numpy

```
In [1]: import numpy as np

In [3]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 2 5 7')
        m[1, 0]

Out[3]: 5
```

Рисунок 2.2 – Элемент матрицы с заданными координатами

```
In [1]: import numpy as np

In [5]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 2 5 7')
        m[1, :]

Out[5]: matrix([[5, 6, 7, 8]])
```

Рисунок 2.3 – Строка матрицы

```
In [6]: m[:, 2]

Out[6]: matrix([[3],
                [7],
                [5]])
```

Рисунок 2.4 – Столбец матрицы

```
In [7]: m[1, 2:]
Out[7]: matrix([[7, 8]])
```

Рисунок 2.5 – Часть строки матрицы

```
In [8]: m[0:2, 1]
Out[8]: matrix([[2],
                [6]])
```

Рисунок 2.6 – Часть столбца матрицы

```
In [9]: m[0:2, 1:3]
Out[9]: matrix([[2, 3],
                [6, 7]])
```

Рисунок 2.7 – Непрерывная часть матрицы

```
In [10]: cols = [0, 1, 3]
         m[:, cols]
Out[10]: matrix([[1, 2, 4],
                 [5, 6, 8],
                 [9, 2, 7]])
```

Рисунок 2.8 – Произвольные столбцы / строки матрицы

```
In [11]: m = np.matrix('1 2 3 4; 5 6 7 8; 9 1 5 7')
         print(m)
[[1 2 3 4]
 [5 6 7 8]
 [9 1 5 7]]

In [13]: type(m)
Out[13]: numpy.matrix
```

Рисунок 2.9 – Создание объект типа matrix.

```
In [14]: m = np.array(m)
         type(m)
Out[14]: numpy.ndarray
```

Рисунок 2.10 – Превращение Matix в ndarray

```
In [16]: m.max()
Out[16]: 9

In [18]: np.max(m)
Out[18]: 9
```

Рисунок 2.10 – Вызов функции расчета статистики

```
In [21]: m.max(axis=1)
Out[21]: matrix([[4],
                 [8],
                 [9]])

In [22]: m.max(axis=0)
Out[22]: matrix([[9, 6, 7, 8]])
```

Рисунок 2.11 – Расчет статистик по строкам или столбцам массива

```
In [23]: m.mean()
Out[23]: 4.833333333333333

In [24]: m.mean(axis=1)
Out[24]: matrix([[2.5],
                 [6.5],
                 [5.5]])

In [25]: m.sum()
Out[25]: 58

In [26]: m.sum(axis=0)
Out[26]: matrix([[15,  9, 15, 19]])
```

Рисунок 2.12 – Функции (методы) для расчета статистик в Numpy

```
In [27]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
         letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])

In [28]: a = True

In [29]: b = 5 > 7
         print(b)

         False

In [30]: less_then_5 = nums < 5
         less_then_5

Out[30]: array([ True,  True,  True,  True, False, False, False, False, False,
                False])

In [31]: pos_a = letters == 'a'
         pos_a

Out[31]: array([ True, False, False, False,  True, False, False])
```

Рисунок 2.13 – Использование boolean массива для доступа к ndarray

```
In [34]: mod_m = np.logical_and(m>=3, m<=7)
         mod_m

Out[34]: matrix([[False, False,  True,  True],
                 [ True,  True,  True, False],
                 [False, False,  True,  True]])

In [35]: m[mod_m]

Out[35]: matrix([[3, 4, 5, 6, 7, 5, 7]])
```

Рисунок 2.14 – Функция logical_and()

```
In [35]: m[mod_m]

Out[35]: matrix([[3, 4, 5, 6, 7, 5, 7]])

In [36]: np.arange(5, 12)

Out[36]: array([ 5,  6,  7,  8,  9, 10, 11])

In [37]: np.arange(10)

Out[37]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [38]: np.arange(1, 5, 0.5)

Out[38]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])
```

Рисунок 2.15 – Функция arrange()

```
In [39]: a = [[1, 2], [3, 4]]
         np.matrix(a)

Out[39]: matrix([[1, 2],
                 [3, 4]])

In [40]: b = np.array([[5, 6], [7, 8]])
         np.matrix(b)

Out[40]: matrix([[5, 6],
                 [7, 8]])

In [41]: np.matrix('[1, 2; 3, 4]')

Out[41]: matrix([[1, 2],
                 [3, 4]])
```

Рисунок 2.16 – Функция np.matrix()

```
In [43]: np.zeros((3, 4))

Out[43]: array([[0., 0., 0., 0.],
                [0., 0., 0., 0.],
                [0., 0., 0., 0.]])

In [44]: np.eye(3)

Out[44]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

Рисунок 2.17 – Функция np.zeros(), np.eye()

```
In [45]: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         A

Out[45]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [46]: np.ravel(A)

Out[46]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [48]: np.ravel(A, order='C')

Out[48]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [49]: np.ravel(A, order='F')

Out[49]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])
```

Рисунок 2.18 – Функция np.ravel()

```
In [50]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
         np.where(a % 2 == 0, a * 10, a / 10)

Out[50]: array([ 0. ,  0.1, 20. ,  0.3, 40. ,  0.5, 60. ,  0.7, 80. ,  0.9])

In [51]: a = np.random.rand(10)
         a

Out[51]: array([0.10534031, 0.8941969 , 0.32697182, 0.16312812, 0.55849046,
                0.98697315, 0.49716072, 0.83889527, 0.00852996, 0.50941908])

In [52]: np.where(a > 0.5, True, False)

Out[52]: array([False,  True, False, False,  True,  True, False,  True, False,
                True])

In [53]: np.where(a > 0.5, 1, -1)

Out[53]: array([-1,  1, -1, -1,  1,  1, -1,  1, -1,  1])
```

Рисунок 2.19 – Функция np.where()

```
In [54]: x = np.linspace(0, 1, 5)
         x

Out[54]: array([0.  , 0.25, 0.5 , 0.75, 1.  ])

In [55]: y =np.linspace(0, 2, 5)
         y

Out[55]: array([0. , 0.5, 1. , 1.5, 2. ])

In [56]: xg, yg = np.meshgrid(x, y)
         xg

Out[56]: array([[0.  , 0.25, 0.5 , 0.75, 1.  ],
                [0.  , 0.25, 0.5 , 0.75, 1.  ],
                [0.  , 0.25, 0.5 , 0.75, 1.  ],
                [0.  , 0.25, 0.5 , 0.75, 1.  ],
                [0.  , 0.25, 0.5 , 0.75, 1.  ]])

In [57]: yg

Out[57]: array([[0. , 0. , 0. , 0. , 0. ],
                [0.5, 0.5, 0.5, 0.5, 0.5],
                [1. , 1. , 1. , 1. , 1. ],
                [1.5, 1.5, 1.5, 1.5, 1.5],
                [2. , 2. , 2. , 2. , 2. ]])
```

Рисунок 2.20 – Функция np.meshgrid()

```
In [17]: np.random.permutation(7)

Out[17]: array([5, 0, 3, 4, 2, 6, 1])

In [18]: a = ['a', 'b', 'c', 'd', 'e']
         np.random.permutation(a)

Out[18]: array(['a', 'd', 'b', 'c', 'e'], dtype='<U1')

In [19]: arr = np.linspace(0, 10, 5)
         arr

Out[19]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

In [20]: arr_mix = np.random.permutation(arr)
         arr_mix

Out[20]: array([ 7.5,  2.5,  0. , 10. ,  5. ])

In [22]: index_mix = np.random.permutation(len(arr_mix))
         index_mix

Out[22]: array([2, 0, 4, 1, 3])

In [23]: arr[index_mix]

Out[23]: array([ 5. ,  0. , 10. ,  2.5,  7.5])
```

Рисунок 2.21 – Функция np.random.pertutation()

2. Индивидуальное задание:

7. Характеристикой столбца целочисленной матрицы назовем сумму модулей его отрицательных нечетных элементов. Переставляя столбцы заданной матрицы, расположить их в соответствий с ростом характеристик. Найти сумму элементов в тех столбцах, которые содержат хотя бы один отрицательный элемент.

Шаг 1: Входной целочисленной матрицы

```
[1]: import numpy as np

[22]: a = np.matrix('2, -5, 6, -7; -33, -17, 9, 2; 45, -23, -54, -1; -3, 7, -93, 6')
      a

[22]: matrix([[  2,  -5,   6,  -7],
              [-33, -17,   9,   2],
              [ 45, -23, -54,  -1],
              [ -3,   7, -93,   6]])
```

Рисунок 2.22 – Результат создания матрицы

Шаг 2: Маска негативных элементов

```
[9]: a < 0

[9]: matrix([[False,  True, False,  True],
             [ True,  True, False, False],
             [False,  True,  True,  True],
             [ True, False,  True, False]])
```

Рисунок 2.23 – Результат маски

Шаг 3: Получаю замаскированные отрицательные элементы из входного массива, используя поэлементное умножение

```
[19]: np.where(a<0,a,0)

[19]: array([[  0,  -5,   0,  -7],
             [-33, -17,   0,   0],
             [  0, -23, -54,  -1],
             [ -3,   0, -93,   0]])
```

Рисунок 2.24 – Результат замаскировки

Шаг 3: Выполняю поэлементное умножение между маской и массивом и получаю суммирование, все за один шаг, используя np.einsum

Что такое np.einsum:

В NumPy мы можем найти Соглашение Эйнштейна о суммировании двух заданных многомерных массивов с помощью numpy.einsum(). Мы передадим два массива в качестве параметра, и он вернет соглашение о суммировании Эйнштейна.

```
[44]: np.einsum('ij,ij->j',a<0,a)

[44]: array([ -36,  -45, -147,   -8])
```

Рисунок 2.25 – Результат использования функции np.einsum()

Шаг 4: Декларирую список с индексами и создание вывода

```
[32]: i = [3, 0, 1, 2]
      new_matrix = a[:,i]
      new_matrix

[32]: matrix([[ -7,   2,  -5,   6],
              [  2, -33, -17,   9],
              [ -1,  45, -23, -54],
              [  6,  -3,   7, -93]])
```

Рисунок 2.26 – Результат перетасовки

Шаг 4: Сумирую все столбци

```
[14]: np.sum(new_matrix, axis=0)

[14]: matrix([[   0,   11,  -38, -132]])
```

Рисунок 2.27 – Результат суммирования

3. Вопросы

1. Каково назначение библиотеки NumPy?
2. Что такое массивы ndarray?
3. Как осуществляется доступ к частям многомерного массива?
4. Как осуществляется расчет статистик по данным?
5. Как выполняется выборка данных из массивов ndarray?