

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Основы работы с библиотекой NumPy»

ОТЧЕТ
по лабораторной работе №3.2
дисциплины
«Технологии распознавания образов»

Выполнил:

Зиёдуллаев Жавохир Эркин угли
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)

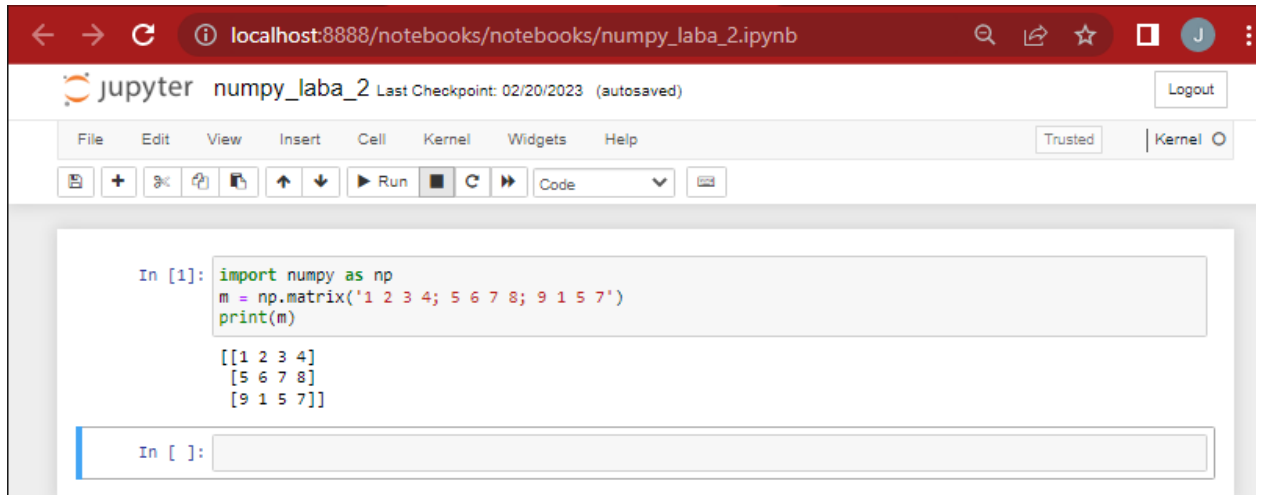
Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Цель: исследовать базовый возможности библиотеки NumPy языка программирования Python

Ссылка: https://github.com/javoxir21/tro_2laba.git

Ход работы:



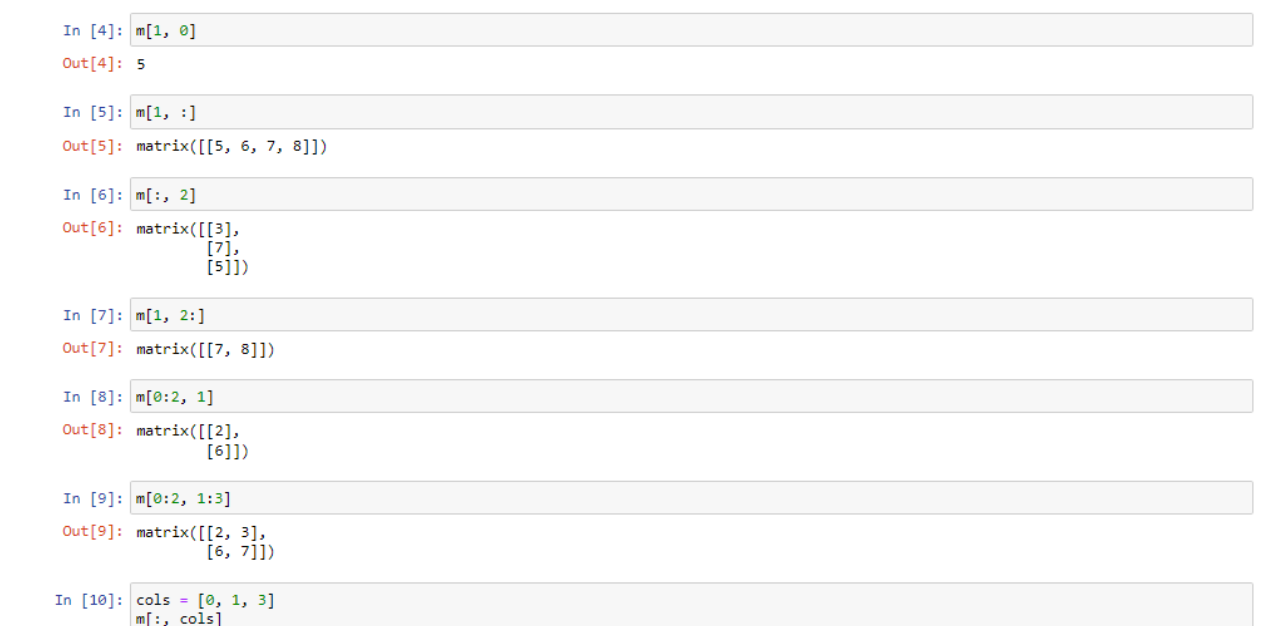
The screenshot shows a Jupyter Notebook running in a web browser at localhost:8888. The notebook is titled 'numpy_laba_2'. The first code cell contains the following Python code:

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

The output of the code is a 3x4 matrix:

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

Рисунок -1 Импортировать библиотека NumPy



The screenshot shows a Jupyter Notebook with several code cells demonstrating indexing operations on a matrix 'm'.

```
In [4]: m[1, 0]
Out[4]: 5
```

```
In [5]: m[1, : ]
Out[5]: matrix([[5, 6, 7, 8]])
```

```
In [6]: m[:, 2]
Out[6]: matrix([[3],
               [7],
               [5]])
```

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

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

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

```
In [10]: cols = [0, 1, 3]
m[:, cols]
```

Рисунок -2 Проработка примеров

```
In [10]: cols = [0, 1, 3]
         m[:, cols]
```

```
Out[10]: matrix([[1, 2, 4],
                 [5, 6, 8],
                 [9, 1, 7]])
```

```
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 [12]: type(m)
```

```
Out[12]: numpy.matrix
```

```
In [13]: m = np.array(m)
         type(m)
```

```
Out[13]: numpy.ndarray
```

```
In [14]: m.shape
```

```
Out[14]: (3, 4)
```

```
In [15]: m.max()
```

```
Out[15]: 9
```

Рисунок -3 Проработка примеров

```
In [23]: m.max()
```

```
Out[23]: 9
```

```
In [24]: m.max(axis=1)
```

```
Out[24]: array([4, 8, 9])
```

```
In [25]: m.max(axis=0)
```

```
Out[25]: array([9, 6, 7, 8])
```

```
In [26]: m.mean()
```

```
Out[26]: 4.833333333333333
```

```
In [27]: m.mean(axis=1)
```

```
Out[27]: array([2.5, 6.5, 5.5])
```

```
In [28]: m.sum()
```

```
Out[28]: 58
```

```
In [29]: m.sum(axis=0)
```

```
Out[29]: array([15, 9, 15, 19])
```

```
In [30]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
         letters = np.array(['a', 'b', 'c', 'd', 'a', 'e', 'b'])
```

```
In [31]: a = True
```

```
In [32]: b = 5 > 7
         print(b)
```

```
False
```

Рисунок -4 Проработка примеров

```

In [33]: less_than_5 = nums < 5
         less_than_5

Out[33]: array([ True,  True,  True,  True, False, False, False, False, False,
                False])

In [34]: pos_a = letters == 'a'
         pos_a

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

In [35]: less_than_5 = nums < 5
         less_than_5

Out[35]: array([ True,  True,  True,  True, False, False, False, False, False,
                False])

In [36]: nums[less_than_5]

Out[36]: array([1, 2, 3, 4])

In [37]: 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 [ ]: |

```

Рисунок -5 Проработка примеров

```

In [38]: mod_m = np.logical_and(m>=3, m<=7)
         mod_m

Out[38]: matrix([[False, False,  True,  True],
                [ True,  True,  True, False],
                [False, False,  True,  True]])

In [39]: m[mod_m]

Out[39]: matrix([[3, 4, 5, 6, 7, 5, 7]])

In [40]: nums = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
         nums[nums < 5]

Out[40]: array([1, 2, 3, 4])

In [41]: nums[nums < 5] = 10
         print(nums)

[10 10 10 10  5  6  7  8  9 10]

In [42]: m[m > 7] = 25
         print(m)

[[ 1  2  3  4]
 [ 5  6  7 25]
 [25  1  5  7]]

```

Рисунок -6 Проработка примеров

```

In [45]: np.arange(10)
Out[45]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [46]: np.arange(5,12)
Out[46]: array([ 5,  6,  7,  8,  9, 10, 11])

In [47]: np.arange(1, 5, 0.5)
Out[47]: array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5])

In [48]: a = [[1, 2], [3, 4]]
          np.matrix(a)
Out[48]: matrix([[1, 2],
                [3, 4]])

In [49]: b = np.array([[5, 6], [7, 8]])
          np.matrix(b)
Out[49]: matrix([[5, 6],
                [7, 8]])

In [50]: np.matrix(['1, 2; 3, 4'])
Out[50]: matrix([[ '1, 2; 3, 4']], dtype='<U10')

In [51]: np.zeros((3, 4))
Out[51]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])

```

Рисунок -7 Проработка примеров

```

In [52]: np.eye(3)
Out[52]: array([[1., 0., 0.],
               [0., 1., 0.],
               [0., 0., 1.]])

In [53]: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
          A
Out[53]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])

In [54]: np.ravel(A)
Out[54]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [55]: np.ravel(A, order='C')
Out[55]: array([1, 2, 3, 4, 5, 6, 7, 8, 9])

In [56]: np.ravel(A, order='F')
Out[56]: array([1, 4, 7, 2, 5, 8, 3, 6, 9])

In [57]: a = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
          np.where(a % 2 == 0, a * 10, 1 / 10)
Out[57]: array([ 0. ,  0.1, 20. ,  0.1, 40. ,  0.1, 60. ,  0.1, 80. ,  0.1])

```

Рисунок -7 Подработка примеров

```

In [58]: a = np.random.rand(10)
a
Out[58]: array([0.4982196 , 0.19489077, 0.01514882, 0.41620375, 0.76445106,
0.04037028, 0.09388995, 0.29658064, 0.76467155, 0.6136983 ])

In [59]: np.where(a > 0.5, True, False)
Out[59]: array([False, False, False, False,  True, False, False, False,  True,
  True])

In [60]: np.where(a > 0.5, 1, -1)
Out[60]: array([-1, -1, -1, -1,  1, -1, -1, -1,  1,  1])

In [61]: x = np.linspace(0, 1, 5)
x
Out[61]: array([0. , 0.25, 0.5 , 0.75, 1.  ])

In [62]: y = np.linspace(0, 2, 5)
y
Out[62]: array([0. , 0.5, 1. , 1.5, 2.  ])

In [63]: xg, yg = np.meshgrid(x, y)
xg
Out[63]: 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.  ]])

```

Рисунок -8 Подработка примеров

```

In [64]: yg
Out[64]: 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. ]])

In [66]: import matplotlib.pyplot as plt
%matplotlib inline

In [67]: plt.plot(xg, yg, color="r", marker="*", linestyle="none")
Out[67]: [<matplotlib.lines.Line2D at 0x1c4fa321b80>,
<matplotlib.lines.Line2D at 0x1c4fa321cd0>,
<matplotlib.lines.Line2D at 0x1c4fa321df0>,
<matplotlib.lines.Line2D at 0x1c4fa321f10>,
<matplotlib.lines.Line2D at 0x1c4fa336070>]

```

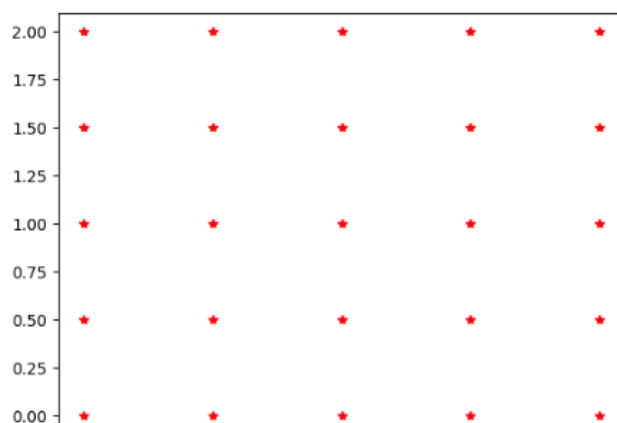
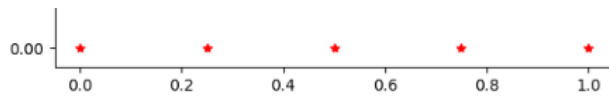


Рисунок -9 Подработка примеров



```
In [68]: np.random.permutation(7)
Out[68]: array([5, 1, 4, 3, 2, 6, 0])

In [69]: a = ['a', 'b', 'c', 'd', 'e']
         np.random.permutation(a)
Out[69]: array(['d', 'e', 'a', 'b', 'c'], dtype='<U1')

In [70]: arr = np.linspace(0, 10, 5)
         arr
Out[70]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

In [71]: arr_mix = np.random.permutation(arr)
         arr_mix
Out[71]: array([ 0. , 10. ,  2.5,  5. ,  7.5])

In [72]: index_mix = np.random.permutation(len(arr_mix))
         index_mix
Out[72]: array([4, 1, 2, 3, 0])

In [73]: arr[index_mix]
Out[73]: array([10. ,  2.5,  5. ,  7.5,  0. ])
```

Рисунок -10 Подработка примеров

Домашние задание

Задание №1

Создайте два массива: в первом должны быть четные числа от 2 до 12 включительно, а в другом числа 7, 11, 15, 18, 23, 29

1. Сложите массивы и возведите элементы получившегося массива в квадрат:

```
In [2]: import numpy as np

         a = np.arange(1,7) * 2
         b = np.array([7, 11, 15, 18, 23, 29])
         print((a + b) ** 2)

[ 81  225  441  676 1089 1681]
```

2. Выведите все элементы из первого массива, индексы которых соответствуют индексам тех элементов второго массива, которые больше 12 и дают остаток 3 при делении на 5.

```
In [3]: a[np.where ((b > 12) & (b % 5 == 3))]
Out[3]: array([ 8, 10])
```

3. Проверьте условие "Элементы первого массива делятся на 4, элементы второго массива меньше 14". (Подсказка: в результате должен получиться массив с True и False)

```
In [4]: (a % 4 == 0) & (b < 14)
Out[4]: array([False,  True, False, False, False, False])
```

Задание №2

Найдите интересный для вас датасет. Например, можно выбрать датасет тут: <http://data.un.org/Explorer.aspx> (выбираете датасет, жмете на view data, потом download, выбирайте csv формат) Рассчитайте подходящие описательные статистики для признаков объектов в выбранном датасете Проанализируйте и прокомментируйте содержательно получившиеся результаты Все комментарии оформляйте строго в ячейках формата markdown

```
In [7]: import csv
import numpy as np
with open('Energy data 1990 - 2020.csv', 'r', newline='', encoding='utf-8') as csvfile:
    reader = csv.reader(csvfile, delimiter=',')
    glob = []
    rus = []
    next(reader)
    for i in reader:
        glob.append(float(i[3]))
        rus.append(float(i[4]))
    stat_glob = np.array(glob)
    stat_rus = np.array(rus)
    print(f"Выбросы от сжигание топливо в мире {np.mean(stat_glob)}, Выбросы от сжигание топливо в России: {np.mean(stat_rus)}")
    print(f"Среднее коэффициент выбросов в мире {np.std(stat_glob)}, Среднее коэффициент выбросов в России: {np.std(stat_rus)}")
    print(f"Интенсивность при неизменном паритете покупательной способности в мире {np.median(stat_glob)}, Интенсивность при неизменном паритете покупательной способности в России: {np.median(stat_rus)}")
    print(f"Общее производства энергии в мире {np.var(stat_glob)}, Общее производства энергии в России: {np.var(stat_rus)}")
```

Выбросы от сжигание топливо в мире 550.3751784030945, Выбросы от сжигание топливо в России: 2.2385420448174487
Среднее коэффициент выбросов в мире 1212.612848530972, Среднее коэффициент выбросов в России: 0.5603883620707869
Интенсивность при неизменном паритете покупательной способности в мире 221.03628085, Интенсивность при неизменном паритете покупательной способности в России: 0.3140351163443793
Общее производства энергии в мире 1470429.9204223985, Общее производства энергии в России: 0.3140351163443793

Ы

Индивидуальное задание:

5. Дана целочисленная квадратная матрица. Определить:

сумму элементов в тех столбцах, которые не содержат отрицательных элементов; минимум среди сумм модулей элементов диагоналей, параллельных побочной диагонали матрицы.

```
In [1]: import numpy as np
n = int(input("Введите размер матрицы (NxN): "))

A = 20*np.random.random(size=(n,n)) - 10
for Row in range(n):
    for Col in range(n):
        print("{0:>5.0f}".format(A[Row][Col]), end=" ")
    print()
print()

summ = 0
isNeg = False

for i in range(n):
    for j in range(n):
        if A[i][j] < 0:
            isNeg = True
            continue
        summ += A[i][j]
    if isNeg == False:
        print("Сумма элементов в столбце без отрицательного элемент: ", summ)
    sum = 0
    isNeg = False

countDiagonal = 2 * n - 1
sumArray = 0
minSum = A[0][n-1]
for i in range(countDiagonal):
    t = n - i - 1
    row = -t if t < 0 else 0
    col = t if t > 0 else 0
    while row < n and col < n:
```



```

while row < n and col < n:
    sumArray += A[row][col]
    row += 1
    col += 1
if minSum > sumArray:
    minSum = sumArray
sumArray = 0
print("Минимум, среды су=умм диагоналей параллельных побочной: ", minSum)

```

Введите размер матрицы (NxN): 4

```

-5  -7  -5  0
-7  -0  -8  -10
9   -8   2  -7
7   -7   1   4

```

Минимум, среды су=умм диагоналей параллельных побочной: -21.13272454615847

Индивидуальное задание

Вывод : исследовали базовый возможности библиотеки NumPy языка программирования Python

Ответы на вопрос

1. Каково назначение библиотеки NumPy?

Математические алгоритмы, реализованные на интерпретируемых языках (например, Python), часто работают гораздо медленнее тех же алгоритмов, реализованных на компилируемых языках (например, Фортран, Си, Java). Библиотека NumPy предоставляет реализации вычислительных алгоритмов (в виде функций и операторов), оптимизированные для работы с многомерными массивами. В результате любой алгоритм, который может быть выражен в виде последовательности операций над массивами (матрицами) и реализованный с использованием NumPy, работает так же быстро, как эквивалентный код, выполняемый в MATLAB.

2. Что такое массивы ndarray?

Ndarray-это объект n-мерного **массива**, определенный в numpy, который хранит коллекцию элементов одинакового типа. Другими словами, мы можем определить **ndarray** как коллекцию объектов типа данных (dtype). Доступ к объекту **ndarray** можно получить с помощью индексации, основанной на 0.

3. Как осуществляется доступ к частям многомерного массива?

При размещении элементов многомерных массивов они располагаются в памяти подряд по строкам, т.е. быстрее всего изменяется последний индекс, а медленнее - первый. Такой порядок дает возможность обращаться к любому элементу многомерного массива, используя адрес его начального элемента и только одно индексное выражение.

Например, обращение к элементу `arr2[1][2]` можно осуществить с помощью указателя `ptr2`, объявленного в форме `int *ptr2=arr2[0]` как обращение `ptr2[1*4+2]` (здесь 1 и 2 это индексы используемого элемента, а 4 это число элементов в строке) или как `ptr2[6]`. Заметим, что внешне похожее обращение `arr2[6]` выполнить невозможно так как указателя с индексом 6 не существует.

4. Как осуществляется расчет статистик по данным?

Рассчитаем несколько описательных статистик для ряда (4.1) с помощью пакета STATISTICA. Предполагается, что пакет установлен на Вашем компьютере. Для решения задачи введем в электронную таблицу пакета исходные данные, т. е. ряд 2, 4, 6, 8, 10 как столбец. В электронной таблице пакета этот ряд будет обозначаться как VAR1. В основном меню пакета выбираем опцию «Статистика» (Statistics). После ее активизации в ниспадающем меню выбираем опцию «Основная статистика/Таблицы» (Basic Statistics/Tables).

5. Как выполняется выборка данных из массивов ndarray?

```
wanted_set = set(wanted)
@numpy.vectorize
def selected(elmt): return elmt in wanted_set
# Or: selected = numpy.vectorize(wanted_set.__contains__)
print test[selected(test[:, 1])]
```