Дисциплина «Искусственный интеллект»
Рабочая тетрадь № 2

Цель машинного обучения – анализ данных.

Данные – зарегистрированная информация; представление фактов, понятий или инструкций в форме, приемлемой для общения, интерпретации, или обработки человеком или с помощью автоматических средств (ISO/IEC/IEEE 24765-2010).

Данные в машинном обучении – это представление информации об исследуемой задаче в виде множеств исследуемых объектов и множеств их характеристик, на основе которых строятся модели, разрабатываются подходы, методы и алгоритмы анализа для принятия решений.

Качество данных – важный аспект машинного обучения.

Для Аналитика (Data Scientist, Data Analyst, Data Mining Engineer) очень важно обладать правильными данными, что гарантирует эффективность обработки и построении прогнозов. На рисунке 1 представлены основные требования к данным.

Остановимся на основных этапах решения задач машинного обучения.

Этапы решения задач машинного обучения:

1. Постановка задачи.
2. Сбор и подготовка данных.
3. Предобработка данных и выделение ключевых признаков.
4. Выбор алгоритмов машинного обучения.
5. Обучение модели (моделей).
6. Оценка качества.
7. Эксплуатация модели.

При подготовке данных можно применять следующие операции:

- структурирование – приведение данных к табличному (матричному) виду;
- заполнение пропусков;

- отбор – исключение записей с отсутствующими или некорректными значениями, если нет возможности заполнения и устранения противоречивости;

- нормализация – приведение числовых значений к определенному диапазону, например к диапазону 0...1;

- кодирование – это представление категориальных данных в числовой форме.

1.1. Теоретический материал – Библиотека NumPy

NumPy (**Num**erical**Py**thon) - это библиотека Python с открытым исходным кодом, которая используется практически во всех областях науки и техники. Это универсальный стандарт для работы с числовыми данными в Python.

Если у вас уже есть Python, вы можете установить NumPy с помощью командной строки:

- pip install numpy

Чтобы начать использовать NumPy необходимо импортировать соответствующую библиотеку:

```
import numpy as np
```

Основным объектом NumPy является однородный многомерный массив (в numpy называется numpy.ndarray). Это многомерный массив элементов (обычно чисел), одного типа.

Наиболее важные атрибуты объектов ndarray:

ndarray.ndim - число измерений (чаще их называют "оси") массива.

ndarray.shape - размеры массива, его форма. Это кортеж натуральных чисел, показывающий длину массива по каждой оси. Для матрицы из n строк и m столбов, shape будет (n,m). Число элементов кортежа shape равно ndim.

ndarray.size - количество элементов массива. Очевидно, равно произведению всех элементов атрибута shape.

ndarray.dtype - объект, описывающий тип элементов массива. Можно определить dtype, используя стандартные типы данных Python. NumPy здесь предоставляет целый букет возможностей, как встроенных, например: bool_, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object_, так и возможность определить собственные типы данных, в том числе и составные.

ndarray.itemsize - размер каждого элемента массива в байтах.

ndarray.data - буфер, содержащий фактические элементы массива. Обычно не нужно использовать этот атрибут, так как обращаться к элементам массива проще всего с помощью индексов.

Подробнее о массивах в NumPy можно найти в официальной документации https://numpy.org/doc/stable/user/absolute_beginners.html

1.2.1 Пример

Задача:

Создать массив 5x2. Создать массив 5x2. Вывести все значения массива, значение элемента с индексом (3,1) и второй столбец. Индексация начинается с нуля.

Решение:

```
import numpy as np
x = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]])
print(x)
print(x[3][1])
print(x[1])
```

Ответ:

```
[[ 1  2]
 [ 3  4]
 [ 5  6]
 [ 7  8]
 [ 9 10]]
8
[3 4]
```

1.2.2 Пример

Задача:

Пример. Выполнить следующее:

1. Создать вектор (одномерный массив) размера 10, заполненный нулями.
2. Создать вектор размера 10, заполненный единицами.
3. Создать вектор размера 10, заполненный заданным числом.
4. Создать вектор со значениями от 10 до 19.

Решение:

```
a = np.zeros(10)
b = np.ones(10)
c = np.full(10, 5)
d = np.arange(10, 20)
print(a, "\n", b, "\n", c, "\n", d)
```

Ответ:

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [5 5 5 5 5 5 5 5 5 5]
 [10 11 12 13 14 15 16 17 18 19]
```

1.2.3 Пример

Задача:

Создать массив 10x10 со случайными значениями, найти минимум, максимум и среднее значение.

Решение:

```
Z = np.random.random((10,10))
Zmin, Zmax, Zmean = Z.min(), Z.max(), Z.mean()
print(Zmin, Zmax, Zmean)
```

Ответ:

```
0.005088982209506376 0.9965682260758483 0.47121463269551994
```

1.2.4 Пример

Задача:

Задать матрицу размерности 5 на 5 и поменять 2 строки в матрице местами.

Решение:

```
A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

Ответ:

```
[[ 5  6  7  8  9]
 [ 0  1  2  3  4]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

1.2.5 Пример

Задача:

Выяснить результат следующих выражений:
0 * np.nan
np.nan == np.nan
np.inf > np.nan
np.nan - np.nan
0.3 == 3 * 0.1

Решение:

```
print(0 * np.nan)
print(np.nan == np.nan)
print(np.inf > np.nan)
print(np.nan - np.nan)
print(0.3 == 3 * 0.1)
```

Ответ:

```
nan
False
False
nan
False
```

1.2.6 Пример

Задача:

Отсортировать массив.

Решение:

```
arr = np.array([2, 1, 5, 3, 7, 4, 6, 8])
print(np.sort(arr))
```

Ответ:

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

1.3.1 Задание

Задача:

Создать 8x8 матрицу и заполнить её в шахматном порядке нулями и единицами.

Решение:

```
a = np.array([[(i+j)%2 for i in range(8)] for j in range(8)])
a
```

Ответ:

```
array([[0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0],
       [0, 1, 0, 1, 0, 1, 0, 1],
       [1, 0, 1, 0, 1, 0, 1, 0]])
```

1.3.2 Задание

Задача:

Создать 5x5 матрицу со значениями в строках от 0 до 4. Для создания необходимо использовать функцию arrange.

Решение:

```
a = np.arange(25).reshape(5, 5) % 5
a
```

Ответ:

```
array([[0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4],
       [0, 1, 2, 3, 4]])
```

1.3.3 Задание

Задача:

Создать массив 3x3x3 со случайными значениями.

Решение:

```
a = np.random.random((3,3,3)).round(3)
a
```

Ответ:

```
array([[[0.551, 0.643, 0.344],
        [0.023, 0.849, 0.784],
        [0.957, 0.562, 0.205]],

       [[0.737, 0.918, 0.713],
        [0.672, 0.512, 0.072],
        [0.353, 0.095, 0.131]],

       [[0.522, 0.712, 0.922],
        [0.944, 0.927, 0.302],
        [0.23 , 0.366, 0.071]]])
```

1.3.4 Задание

Задача:

Создать матрицу с 0 внутри, и 1 на границах.

Решение:

```
a = np.array([[1 if (i % 7) == 0 or (j % 7) == 0 else 0 for i in range(8)] for j
in range(8)])
```

```
a
```

Ответ:

```
array([[1, 1, 1, 1, 1, 1, 1, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 0, 0, 0, 0, 0, 0, 1],
       [1, 1, 1, 1, 1, 1, 1, 1]])
```

1.3.5 Задание

Задача:

Создайте массив и отсортируйте его по убыванию.

Решение:

```
a = np.random.randint(100, size=(10, 10))
a = np.sort(a, axis=None).reshape(10, 10)
a = np.flip(a)
a
```

Ответ:

```
array([[99, 99, 97, 94, 93, 93, 92, 92, 92, 89],
       [88, 87, 86, 84, 84, 82, 81, 79, 77, 76],
       [74, 72, 70, 70, 69, 68, 68, 68, 67, 66],
       [64, 63, 62, 62, 60, 60, 60, 60, 59, 58],
       [58, 55, 54, 54, 53, 52, 51, 50, 50, 49],
       [47, 44, 43, 43, 43, 43, 40, 38, 38, 37],
       [35, 32, 31, 29, 29, 28, 28, 27, 26, 26],
       [24, 24, 23, 23, 22, 22, 22, 19, 19, 17],
       [17, 17, 16, 15, 13, 13, 12, 11, 10,  9],
       [ 9,  8,  8,  7,  7,  4,  1,  1,  1,  0]])
```

1.3.6 Задание

Задача:

Создайте матрицу, выведите ее форму, размер и размерность.

Решение:

```
a = np.random.randint(100, size=(10, 10))
print(a.shape)
print(a.size)
```

```
print(a.ndim)
```

Ответ:

```
(10, 10)
100
2
```

2.1. Теоретический материал – Библиотека Pandas

Первым шагом в любом начинании в области машинного обучения является введение исходных данных в систему. Исходные данные могут вводиться вручную, содержаться в файле или храниться в интернете в каком-либо формате. Кроме того, часто требуется получить данные из нескольких источников.

Библиотека pandas – это удобный и быстрый инструмент для работы с данными, обладающий большим функционалом. Если очень кратко, то *pandas* – это библиотека, которая предоставляет очень удобные с точки зрения использования инструменты для хранения данных и работе с ними.

Библиотека pandas присутствует в стандартной поставке Anaconda. Если же ее там нет, то его можно установить отдельно. Для этого введите командной строке:

- pip install pandas

Для импорта библиотеки используйте команду:

```
import pandas as pd
```

Библиотека pandas предоставляет две ключевые структуры данных: Series и DataFrame.

Series – это одномерная структура данных, ее можно представить, как таблицу с одной строкой. С Series можно работать как с обычным массивом (обращаться по номеру индекса), и как с ассоциированным массивом, когда можно использовать ключ для доступа к элементам данных.

DataFrame – это двумерная структура. Идейно она очень похожа на обычную таблицу, что выражается в способе ее создания и работе с ее элементами.

2.2.1 Пример

Задача:

Создать Series из списка Python, словаря Python, и массива Numpy (установить буквенные метки для последнего).

Решение:

```
import pandas as pd
lst = [1, 2, 3, 4, 5]
d = {'a':1, 'b':2, 'c':3}
ndarr = np.array([1, 2, 3, 4, 5])

s1 = pd.Series(lst)
s2 = pd.Series(d)
s3 = pd.Series(ndarr, ['a', 'b', 'c', 'd', 'e'])

print(s1)
print(s2)
print(s3)
```

Ответ:

```
0    1
1    2
2    3
3    4
4    5
dtype: int64
a    1
b    2
c    3
dtype: int64
a    1
b    2
c    3
d    4
e    5
dtype: int32
```

2.2.2 Пример

Задача:

Дано два Series. Напечатать их первые элементы и все элементы после третьего (во втором фрейме).

Решение:

```
s1 = pd.Series([1, 2, 3, 4, 5], ['a', 'b', 'c', 'd', 'e'])
s2 = pd.Series([5, 4, 3, 2, 1])
print(s1['a'])
print(s2[0])
print(s2[3:])
```

Ответ:

```
1
5
3    2
4    1
dtype: int64
```

2.2.3 Пример

Задача:

Создайте новый фрейм данных.

Решение:

```
dataframe = pd.DataFrame()
dataframe['Имя'] = ['Джеки Джексон', 'Стивен Стивенсон']
dataframe['Возраст'] = [38, 25]
dataframe['Водитель'] = [True, False]
dataframe
```

Ответ:

	Имя	Возраст	Водитель
0	Джеки Джексон	38	True
1	Стивен Стивенсон	25	False

2.2.4 Пример

Задача:

Загрузите фрейм данных по ссылке: https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv

Решение:

```
# Создать URL-адрес
url = 'https://raw.githubusercontent.com/chrisalbon/simulated_datasets/master/titanic.csv'
# Загрузить данные
dataframe = pd.read_csv(url)
# Показать пять строк
dataframe.head(5)
```

Ответ:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.00	female	1	1
1	Allison, Miss Helen Loraine	1st	2.00	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.00	male	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	25.00	female	0	1
4	Allison, Master Hudson Trevor	1st	0.92	male	1	0

2.2.5 Пример

Задача:

Пронализировать характеристики фрейма данных.

Решение:

Одна из самых простых вещей, которые мы можем сделать после загрузки данных, – это взглянуть на первые несколько строк с помощью метода head. На последние строки можно посмотреть с помощью функции tail. Мы также можем взглянуть на количество строк и столбцов: dataframe.shape. Кроме того, используя метод describe, мы

можем получить описательную статистику для любых числовых столбцов.

```
dataframe.head(2)
dataframe.tail(3)
dataframe.shape
dataframe.describe()
```

Более подробно с возможностями работы с фреймами данных можно узнать по ссылке ниже: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Ответ:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

	Name	PClass	Age	Sex	Survived	SexCode
1310	Zenni, Mr Philip	3rd	22.0	male	0	0
1311	Lievens, Mr Rene	3rd	24.0	male	0	0
1312	Zimmerman, Leo	3rd	29.0	male	0	0

```
(1313, 6)
```

	Age	Survived	SexCode
count	756.000000	1313.000000	1313.000000
mean	30.397989	0.342727	0.351866
std	14.259049	0.474802	0.477734
min	0.170000	0.000000	0.000000
25%	21.000000	0.000000	0.000000
50%	28.000000	0.000000	0.000000
75%	39.000000	1.000000	1.000000
max	71.000000	1.000000	1.000000

2.2.6 Пример

Задача:

Выберите индивидуальные данные или срезы фрейма данных.

Решение:

Для выбора одной или нескольких строк, либо значений, можно использовать методы **1oc** или **iloc.**

```
dataframe.iloc[1:4]
```

Ответ:

	Name	PClass	Age	Sex	Survived	SexCode
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1
2	Allison, Mr Hudson Joshua Creighton	1st	30.0	male	0	0
3	Allison, Mrs Hudson JC (Bessie Waldo Daniels)	1st	25.0	female	0	1

2.2.7 Пример

Задача:

Требуется отобрать строки фрейма данных на основе некоторого условия. Необходимо сформировать новый фрейм данных из пассажиров первого класса.

Решение:

```
dataframe[dataframe['PClass'] == '1st'].head(2)
```

Ответ:

	Name	PClass	Age	Sex	Survived	SexCode
0	Allen, Miss Elisabeth Walton	1st	29.0	female	1	1
1	Allison, Miss Helen Loraine	1st	2.0	female	0	1

2.3.1 Задание

Задача:

Найдите евклидово расстояние между двумя Series (точками) a и b, не используя встроенную формулу.

Решение:

```
import math
a = pd.Series([25, 13, 44])
b = pd.Series([834, 91, 9])
s = np.sqrt(sum(pow(a[i]-b[i], 2) for i in range(len(a))))
print(s)
distance = math.dist(a, b) # проверка
print(distance)
```

Ответ:

```
813.5047633542166
813.5047633542166
```

2.3.2 Задание

Задача:

Найдите в Интернете ссылку на любой csv файл и сформируйте из него фрейм данных (например, коллекцию фреймов данных можно найти здесь: https://github.com/akmand/datasets).

Решение:

```
df = pd.read_csv('titanic.csv')
df.head(3)
```

Ответ:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

2.3.3 Задание

Задача:

Проделайте с получившемся из предыдущего задания фреймом данных те же действия, что и в примерах 2.2.5-2.2.7.

Решение:

```
df.info()
df.shape
df.describe()
df.loc[::5]
df[df['Age'] > 30]
```

Ответ:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
(891, 12)
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000
```

```
df.loc[::5]
✓ 0.0s                                                                    Python
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
870	871	0	3	Balkic, Mr. Cerin	male	26.0	0	0	349248	7.8958	NaN	S
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	female	15.0	0	0	2667	7.2250	NaN	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	female	25.0	0	1	230433	26.0000	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	female	39.0	0	5	382652	29.1250	NaN	Q
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

179 rows × 12 columns

```
df[df['Age'] > 30].head(3)
✓ 0.0s                                                                    Python
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

3.1. Теоретический материал – Работа с числовыми данными

Количественные данные что-то измеряют – будь то размер класса, ежемесячные продажи или оценки учащихся. Естественным способом представления этих величин является численным (например, 150 студентов, $529 392 продаж).

Нормализация данных — это общепринятая задача предобработки в машинном обучении. Многие алгоритмы предполагают, что все признаки находятся в единой шкале, как правило, от 0 до 1 или от -1 до 1.

Существует множество способов нормализации значений признаков, чтобы масштабировать их к единому диапазону и использовать в различных моделях машинного обучения. В зависимости от используемой функции, их можно разделить на 2 большие группы: линейные и нелинейные. При нелинейной нормализации в расчетных соотношениях используются функции логистической сигмоиды или гиперболического тангенса. В линейной нормализации изменение переменных осуществляется пропорционально, по линейному закону.

На практике наиболее распространены следующие методы нормализации признаков:

- ***Минимакс*** – линейное преобразование данных в диапазоне [0..1], где минимальное и максимальное масштабируемые значения соответствуют 0 и 1 соответственно;
- ***Z-масштабирование*** данных на основе среднего значения и стандартного отклонения: производят деление разницы между переменной и средним значением на стандартное отклонение.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

нормализация по методу **минимакс**

$$z = \frac{x - \mu}{\sigma}$$

Z-масштабирование

При масштабировании данных мы будем использовать одну из популярных библиотек машинного обучения *Scikit-learn*. Библиотека содержит пакет sklearn.preprocessing, который предоставляет широкие возможности для нормализации данных. Следует отметить, что в целом алгоритмы обучения выигрывают от стандартизации набора данных.

3.2.1. Пример

Задача:

Прошкалируйте числовой признак в диапазон между двумя значениями.

Решение:

```
# Загрузить библиотеки
import numpy as np
from sklearn import preprocessing

# Создать признак
feature = np.array([[-500.5], [-100.1], [0], [100.1], [900.9]])
# Создать шкалировщик
minmax_scale = preprocessing.MinMaxScaler(feature_range = (0, 1))

# Прошкалировать признак
scaled_feature = minmax_scale.fit_transform(feature)

#Показать прошкалированный признак
scaled_feature
```

Ответ:

```
array([[0.        ],
       [0.28571429],
       [0.35714286],
       [0.42857143],
       [1.        ]])
```

3.2.2. Пример

Задача:

Преобразуйте признак, чтобы он имел среднее значение 0 и стандартное отклонение 1.

Решение:

```
x = np.array([[-1000.1], [-200.2], [500.5], [600.6], [9000.9]])
# Создать шкалировщик
scaler = preprocessing.StandardScaler()
# Преобразовать признак
standardized = scaler.fit_transform(x)
# Показать признак
standardized
```

Мы можем увидеть эффект стандартизации, обратившись к среднему значению и стандартному отклонению результата нашего решения:

```
print("Среднее:", round(standardized.mean()))
print("Стандартное отклонение:", standardized.std())
```

Ответ:

```
array([[-0.76058269],
       [-0.54177196],
       [-0.35009716],
       [-0.32271504],
       [ 1.97516685]])

Среднее: 0
Стандартное отклонение: 1.0
```

3.2.3. Пример

Задача:

Дан фрейм данных

```
dfTest = pd.DataFrame({'A':[14.00,90.20,90.95,96.27,91.21],
                       'B':[103.02,107.26,110.35,114.23,114.68],
                       'C':['big','small','big','small','small']})
```

Необходимо масштабировать его числовые столбцы.

Решение:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
dfTest = pd.DataFrame({'A':[14.00,90.20,90.95,96.27,91.21],
                       'B':[103.02,107.26,110.35,114.23,114.68],
                       'C':['big','small','big','small','small']})

dfTest[['A', 'B']] = scaler.fit_transform(dfTest[['A', 'B']])
dfTest
```

Ответ:

	A	B	C
0	0.000000	0.000000	big
1	0.926219	0.363636	small
2	0.935335	0.628645	big
3	1.000000	0.961407	small
4	0.938495	1.000000	small

3.3.2 Задание

Задача:

Загрузить фрейм данных по ссылке: https://raw.githubusercontent.com/akmand/datasets/master/iris.csv. Необходимо выполнить нормализацию первого числового признака (sepal_length_cm) с использованием минимаксного преобразования, а второго (sepal_width_cm) с задействованием z-масштабирования.

Решение:

```
iris[['sepal.length']] =
MinMaxScaler().fit_transform(iris[['sepal.length']])
iris['sepal.length']
iris[['sepal.width']] = stats.zscore(iris[['sepal.width']])
iris[['sepal.width']]
```

Ответ:

```
0      0.222222
1      0.166667
2      0.111111
3      0.083333
4      0.194444
         ...
145    0.666667
146    0.555556
147    0.611111
148    0.527778
149    0.444444
```

	sepal.width
0	1.019004
1	-0.131979
2	0.328414
3	0.098217
4	1.249201
...	...
145	-0.131979
146	-1.282963
147	-0.131979
148	0.788808
149	-0.131979