Цель и требования к данным в машинном обучении

Цель машинного обучения — анализ данных.

Данные — это зарегистрированная информация; представление фактов, понятий или инструкций в форме, приёмлемой 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) крайне важно обладать правильными данными, что гарантирует эффективность обработки и построения прогнозов.

Список ключевых требований к качеству данных в машинном обучении:

- **Доступность** — данные должны быть доступны для анализа и использования
- **Точность** — значения должны быть корректными и свободными от ошибок
- **Полнота** — отсутствие пропущенных или неполных значений
- **Непротиворечивость** — отсутствие конфликтующих записей
- **Однозначность** — каждое значение должно иметь чёткое и единое толкование
- **Релевантность** — данные должны быть связаны с поставленной задачей
- **Надежность** — информация должна быть получена из достоверных источников
- **Своевременность** — данные должны быть актуальными на момент анализа
- **Взаимосвязанность** — наличие связей между объектами и характеристиками

Этапы решения задач машинного обучения

Остановимся на основных этапах решения задач машинного обучения.

Основные этапы:

1. **Постановка задачи**
   Чёткое определение цели: что нужно предсказать, какая метрика важна, какие данные доступны.

2. **Сбор и подготовка данных**
   Получение исходной информации из различных источников (базы, API, файлы и т.д.).

3. **Предобработка данных и выделение ключевых признаков**
   Подготовка данных к обучению модели — очистка, преобразование и выбор релевантных характеристик.

4. **Выбор алгоритмов машинного обучения**
   Подбор подходящих моделей (регрессия, классификация, кластеризация и др.) в зависимости от типа задачи.

5. **Обучение модели (моделей)**
   Настройка параметров модели на обучающей выборке.

6. **Оценка качества**
   Проверка производительности модели на тестовой выборке с помощью метрик (например, accuracy, MSE, F1 и др.).

7. **Эксплуатация модели**
   Использование обученной модели для прогнозирования на новых данных.

Операции при подготовке данных:

- **Структурирование** — приведение данных к табличному (матричному) виду
- **Заполнение пропусков** — восстановление отсутствующих значений (медиана, среднее, интерполяция и т.д.)
- **Отбор** — исключение записей с отсутствующими или некорректными значениями, если невозможно их заполнить или устранить противоречивость
- **Нормализация** — приведение числовых значений к определённому диапазону, например, $[0; 1]$ или $[-1; 1]$
- **Кодирование** — представление категориальных данных в числовой форме (например, one-hot encoding, label encoding)

1.1. Теоретический материал – Библиотека NumPy

NumPy (**Numerical Python**) — это библиотека Python с открытым исходным кодом, используемая практически во всех областях науки и техники.
Она является универсальным стандартом для работы с числовыми данными в Python.

Установка

Если у вас уже установлен Python, вы можете установить NumPy через командную строку:

```
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 размером 5×2.
Вывести:

- все значения массива,
- значение элемента с индексом $(3,1)$,
- второй столбец.

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:

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 (например, 5).
4. Создать вектор со значениями от 10 до 19.

Решение:

```
import numpy as np

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 размером 10×10 со случайными значениями из интервала $[0; 1)$, найти:

- минимальное значение,
- максимальное значение,
- среднее значение.

Решение:

```
import numpy as np

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 5×5 и поменять местами 2 строки в матрице.

Решение:

```
import numpy as np
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:

```
import numpy as np

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:

```
import numpy as np

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 матрицу размером 8×8 и заполнить её в шахматном порядке нулями и единицами.

Решение:

In [3]:
```
import numpy as np

n = 8
matrix = np.zeros((n, n), dtype=int)

for i in range(n):
    for j in range(n):
        if (i + j) % 2 == 0:
            matrix[i, j] = 1
print(matrix)
```

```
[[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]
 [0 1 0 1 0 1 0 1]]
```

1.3.2. Задание

Задача:

Создать матрицу размером 5×5, где значения в каждой строке изменяются от 0 до 4 (включительно).
Для создания необходимо использовать функцию `np.arange()`.

Решение:

In [5]:
```
import numpy as np

row = np.arange(5)
matrix = np.tile(row, (5, 1))
print(matrix)
```

```
[[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 трёхмерный массив размером $3 \times 3 \times 3$ со случайными значениями.

Решение:

In [7]:
```
import numpy as np

arr = np.random.random((3, 3, 3))
print(arr)
```

```
[[[0.84656891 0.83926961 0.50902257]
  [0.91487627 0.97384012 0.57987084]
  [0.80560515 0.29127412 0.60842649]]

 [[0.71395718 0.14073265 0.16936943]
  [0.79532607 0.2259461  0.79088918]
  [0.60964739 0.83822706 0.23646533]]

 [[0.07959279 0.73225294 0.18468675]
  [0.21131302 0.94541392 0.11249589]
  [0.10994764 0.16625827 0.84601594]]]
```

1.3.4. Задание

Задача:

Создать матрицу, где:

- внутри — значение `0`,
- на границах — значение `1`.

Решение:

In [8]:
```
import numpy as np

n = int(input())
m = int(input())
A = np.zeros((n, m))
for i in range(0, len(A[0])):
  A[0, i] = 1
  A[len(A)-1, i] = 1
for i in range(0, len(A)):
  A[i, 0] = 1
  A[i, len(A[0]) - 1] = 1
print(A)
```

```
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

1.3.5. Задание

Задача:

Создайте массив и отсортируйте его по убыванию.

Решение:

In [9]:
```
import numpy as np

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

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

1.3.6. Задание

Задача:

Создайте матрицу, выведите её:

- форму (`shape`),
- размер (`size`),
- размерность (`ndim`).

Решение:

In [12]:
```
import numpy as np

matrix = np.array([[1, 2, 3, 4],
                   [5, 6, 7, 8],
                   [9, 10, 11, 12]])

print("Форма (shape):", matrix.shape)
print("Размер (size):", matrix.size)
print("Размерность (ndim):", matrix.ndim)
```

```
Форма (shape): (3, 4)
Размер (size): 12
Размерность (ndim): 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 файлах (CSV, Excel и др.) или храниться в интернете в каком-либо формате.
Часто требуется объединить 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 объект `Series` из:

- списка Python,
- словаря Python,
- массива NumPy (с установкой буквенных меток индексов).

Решение:

```
import pandas as pd
import numpy as np

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: int64
```

2.2.2. Пример

Задача:

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

Решение:

```
import pandas as pd

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 (`DataFrame`) с информацией о двух людях:

Решение:

```
import pandas as pd

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

```
dataframe['Водитель'] = [True, False]
print(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:

```
import pandas as pd

url = 'https://raw.githubusercontent.com/chrisalbon/
simulated_datasets/master/titanic.csv'
dataframe = pd.read_csv(url)
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:

Проанализировать 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 29.00 female 1 1
1 Allison, Miss Helen Loraine 1st 2.00 female 0 1
---
Name PClass Age Sex Survived SexCode
1310 Zenni, Mr Philip 3rd 22.00 male 0 0
1311 Lievens, Mr Rene 3rd 24.00 male 0 0
1312 Zimmerman, Leo 3rd 29.00 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 | 2.00|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
```

2.2.7. Пример

Задача:

Отобрать строки фрейма данных на основе условия:
сформировать новый фрейм из пассажиров **первого класса**.

Решение:

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

Ответ:

```
                 Name              PClass   Age     Sex       Survived
SexCode
0  Allen, Miss Elisabeth Walton      1st   29.00  female         1
1
1  Allison, Miss Helen Loraine       1st    2.00  female         0
1
```

2.3.1. Задание

Задача:

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

Решение:

In [19]:
```
import pandas as pd
import numpy as np

a = pd.Series([2, 4, 6, 8])
b = pd.Series([1, 3, 5, 7])

dist = np.sqrt(np.sum((a - b)**2))

print(dist)
```

```
2.0
```

2.3.2. Задание

Задача:

Найдите в Интернете ссылку на любой CSV-файл и сформируйте из него фрейм данных.

> - https://github.com/akmand/datasets — коллекция открытых наборов данных

Решение:

In [20]:
```
import pandas as pd
import numpy as np

url = 'https://raw.githubusercontent.com/akmand/datasets/main/FMLPDA_Table4_3.
dataframe = pd.read_csv(url)
print(dataframe.head())
```

```
  stream     slope elevation vegetation
0  False     steep      high  chapparal
1   True  moderate       low   riparian
2   True     steep    medium   riparian
3  False     steep    medium  chapparal
4  False      flat      high    conifer
```

2.3.3. Задание

Задача:

Проделайте с получившимся из предыдущего задания фреймом данных те же действия, что и в примерах **2.2.5–2.2.7**.

Решение:

In [30]:
```
import pandas as pd
import numpy as np

url = 'https://raw.githubusercontent.com/akmand/datasets/main/FMLPDA_Table4_3.
dataframe = pd.read_csv(url)
print(dataframe)
print("\n", '2.2.5')
print(dataframe.head(2))
print("\n")
print(dataframe.tail(3))
print("\n")
print(dataframe.shape)
print("\n")
print(dataframe.describe())
print("\n", '2.2.6')
print(dataframe.iloc[2:4])
print("\n", '2.2.7')
print(dataframe[dataframe['slope'] == 'steep'].head())
```

```
   stream     slope elevation vegetation
0   False     steep      high  chapparal
1    True  moderate       low   riparian
2    True     steep    medium   riparian
3   False     steep    medium  chapparal
4   False      flat      high    conifer
5    True     steep   highest    conifer
6    True     steep      high  chapparal

 2.2.5
   stream     slope elevation vegetation
0   False     steep      high  chapparal
1    True  moderate       low   riparian


   stream  slope elevation vegetation
4   False   flat      high    conifer
5    True  steep   highest    conifer
6    True  steep      high  chapparal


(7, 4)


       stream  slope elevation vegetation
count       7      7         7          7
unique      2      3         4          3
top      True  steep      high  chapparal
freq        4      5         3          3

 2.2.6
   stream  slope elevation vegetation
2    True  steep    medium   riparian
3   False  steep    medium  chapparal

 2.2.7
   stream  slope elevation vegetation
0   False  steep      high  chapparal
2    True  steep    medium   riparian
3   False  steep    medium  chapparal
5    True  steep   highest    conifer
6    True  steep      high  chapparal
```

3.1. Теоретический материал – Работа с числовыми данными

Количественные данные — это результат измерений: размер класса, ежемесячные продажи, оценки учащихся и т.д.
Естественным способом представления таких величин является численный формат (например, 150 студентов, $529\,392$ продаж).

Нормализация данных

Нормализация — это общепринятая задача предобработки в машинном обучении.
Многие алгоритмы требуют, чтобы все признаки находились в единой шкале, например:

- от 0 до 1, или
- от -1 до 1

Это помогает ускорить обучение моделей и улучшить их стабильность.

Существует множество способов нормализации. В зависимости от функции преобразования, они делятся на две группы:

- **линейные** — изменение пропорционально (по линейному закону)
- **нелинейные** — используются функции, такие как сигмоида или гиперболический тангенс

На практике наиболее распространены следующие методы нормализации:

🔹 Минимакс-нормализация (Min-Max Scaling)

Линейное преобразование данных в диапазоне $[0; 1]$.
Формула: $$ x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} $$

🔹 Z-масштабирование (Z-Score Normalization)

Преобразование на основе среднего значения и стандартного отклонения.
Формула: $$ x_{\text{z}} = \frac{x - \mu}{\sigma} $$ где:

- μ — среднее значение,
- σ — стандартное отклонение.

> 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:

Прошколируйте числовой признак в диапазон между двумя значениями (например, от 0 до 1).

Решение:

```
import numpy as np
from sklearn import preprocessing

# Создание признака
feature = np.array([[-500.5], [-100.1], [0], [100.1], [900.9]])

# Создание нормализатора: масштабирование в диапазон [0, 1]
minmax_scale = preprocessing.MinMaxScaler(feature_range=(0, 1))

# Прошкалирование признака
scaled_feature = minmax_scale.fit_transform(feature)

# Вывод результата
print(scaled_feature)
```

Ответ:

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

3.2.2. Пример

Задача:

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

Решение:

```
import numpy as np
from sklearn import preprocessing

x = np.array([[-100.1], [-200.2], [500.5], [600.6], [900.9]])
scaler = preprocessing.StandardScaler()
standardized = scaler.fit_transform(x)
print(standardized)
```

Ответ:

```
array([[-0.76058269]
       [-0.54177196]
       [-0.35009716]
       [-0.32271504]
       [ 1.97516685]])
Среднее: 0.0
Стандартное отклонение: 1.0
```

3.2.3. Пример

Задача:

Дан фрейм данных с числовыми и категориальными признаками:

```
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 только числовые столбцы.

Решение:

```
import pandas as pd
from sklearn.preprocessing import 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']
})

scaler = MinMaxScaler()
dfTest[['A', 'B']] = scaler.fit_transform(dfTest[['A', 'B']])
print(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** (Min-Max Scaling),
- второго числового признака (`sepal_width_cm`) — с использованием **Z-масштабирования** (StandardScaler).

Решение:

In [29]:
```
import pandas as pd
from sklearn import preprocessing

url = 'https://raw.githubusercontent.com/akmand/datasets/master/iris.csv'
dataframe = pd.read_csv(url)

min_max_scaler = preprocessing.MinMaxScaler()
dataframe['sepal_length_cm'] = min_max_scaler.fit_transform(dataframe[['sepal_

z_score_scaler = preprocessing.StandardScaler()
dataframe['sepal_width_cm'] = z_score_scaler.fit_transform(dataframe[['sepal_w

print(dataframe['sepal_length_cm'], "\n")
print(dataframe['sepal_width_cm'])
```

```
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
Name: sepal_length_cm, Length: 150, dtype: float64

0      1.032057
1     -0.124958
2      0.337848
3      0.106445
4      1.263460
         ...
145   -0.124958
146   -1.281972
147   -0.124958
148    0.800654
149   -0.124958
Name: sepal_width_cm, Length: 150, dtype: float64
```

In [ ]: