

**Московский государственный технический  
университет им. Н. Э. Баумана**

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №1

Выполнил:  
Каятский П. Е.  
группа ИУ5-64Б

Проверил:  
Гапанюк Ю.Е.

Дата: 07.04.25

Дата:

Подпись:

Подпись:

Москва, 2025 г.

**Цель лабораторной работы:** изучение различных методов визуализация данных.

**Краткое описание.** Построение основных графиков, входящих в этап разведочного анализа данных.

**Задание:**

- Выбрать набор данных (датасет). Вы можете найти список свободно распространяемых датасетов [здесь](#).
- Для первой лабораторной работы рекомендуется использовать датасет без пропусков в данных, например из [Scikit-learn](#).
- Пример преобразования датасетов Scikit-learn в Pandas Dataframe можно посмотреть [здесь](#).

Для лабораторных работ не рекомендуется выбирать датасеты большого размера.

- Создать ноутбук, который содержит следующие разделы:
  1. Текстовое описание выбранного Вами набора данных.
  2. Основные характеристики датасета. (Мой датасет: <https://www.kaggle.com/datasets/muhammetvarl/laptop-price>)
  3. Визуальное исследование датасета.
  4. Информация о корреляции признаков.
- Сформировать отчет и разместить его в своем репозитории на github.

Средства и способы визуализации данных можно посмотреть [здесь](#).

В качестве опорного примера для выполнения лабораторной работы можно использовать [пример](#).

Дополнительно примеры решения задач, содержащие визуализацию, можно посмотреть в репозитории курса mlcourse.ai  
- [https://github.com/Yorko/mlcourse.ai/wiki/Individual-projects-and-tutorials-\(in-Russian\)](https://github.com/Yorko/mlcourse.ai/wiki/Individual-projects-and-tutorials-(in-Russian))

**Ход выполнения:**

# Разведочный анализ данных. Исследование и визуализация данных.

## Импорт библиотек

Импортируем библиотеки с помощью команды `import`. Как правило, все команды `import` размещают в первой ячейке ноутбука, но мы в этом примере будем подключать все библиотеки последовательно, по мере их использования.

```
[1] 1 > import ...
5 %matplotlib inline
6 sns.set(style="ticks")
Executed at 2025.02.20 12:36:34 in 19ms
```

## Загрузка данных

Загрузим файлы датасета в помощью библиотеки Pandas.

Не смотря на то, что файлы имеют расширение `txt` они представляют собой данные в формате CSV (<https://ru.wikipedia.org/wiki/CSV>). Часто в файлах такого формата в качестве разделителей используются символы `;`, `;` или табуляция. Поэтому вызывая метод `read_csv` всегда стоит явно указывать разделитель данных с помощью параметра `sep`. Чтобы узнать какой разделитель используется в файле его рекомендуется предварительно посмотреть в любом текстовом редакторе.

```
[2] 1 # Будем анализировать данные только на обучающей выборке
2 data = pd.read_csv('laptop_price - dataset.csv', sep=";")
Executed at 2025.02.20 12:36:40 in 32ms
```

## 2) Основные характеристики датасета

```
[3] 1 # Первые 5 строк датасета
2 data.head()
Executed at 2025.02.20 12:37:41 in 23ms
```

	Company	Product	TypeName	Inches	ScreenResolution	CPU_Company	CPU_
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel	Core
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel	Core
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel	Core
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel	Core
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel	Core

```
[4] 1 # Размер датасета - 1275 строк, 15 колонок
2 data.shape
Executed at 2025.02.20 12:37:54 in 7ms
```

(1275, 15)

```
[5] 1 total_count = data.shape[0]
2 print('Всего строк: {}'.format(total_count))
Executed at 2025.02.20 12:38:13 in 5ms
```

Всего строк: 1275

```
[6] 1 # Список колонок
2 data.columns
Executed at 2025.02.20 12:38:22 in 8ms

Index(['Company', 'Product', 'TypeName', 'Inches', 'ScreenResolution',
      'CPU_Company', 'CPU_Type', 'CPU_Frequency (GHz)', 'RAM (GB)', 'Memory',
      'GPU_Company', 'GPU_Type', 'OpSys', 'Weight (kg)', 'Price (Euro)'],
      dtype='object')

[10] 1 # Список колонок с типами данных
2 data.dtypes
Executed at 2025.02.20 12:39:39 in 7ms

CPU_Company      object
CPU_Type          object
CPU_Frequency (GHz) float64
RAM (GB)          int64
Memory           object
GPU_Company      object
GPU_Type         object
OpSys            object
Weight (kg)      float64
Price (Euro)     float64
dtype: object
```

```
[9] 1 # Проверим наличие пустых значений
2 # Цикл по колонкам датасета
3 for col in data.columns:
4     # Количество пустых значений - все значения заполнены
5     temp_null_count = data[data[col].isnull()].shape[0]
6     print('{} - {}'.format(col, temp_null_count))
Executed at 2025.02.20 12:39:31 in 15ms

ScreenResolution - 0
CPU_Company - 0
CPU_Type - 0
CPU_Frequency (GHz) - 0
RAM (GB) - 0
Memory - 0
GPU_Company - 0
GPU_Type - 0
OpSys - 0
Weight (kg) - 0
Price (Euro) - 0
```

```
[11] 1 # Основные статистические характеристики набора данных
2 data.describe()
Executed at 2025.02.20 12:40:27 in 16ms
```

	Inches	CPU_Frequency (GHz)	RAM (GB)	Weight (kg)	Price (Euro)
count	1275.000000	1275.000000	1275.000000	1275.000000	1275.000000
mean	15.022902	2.302980	8.440784	2.040525	1134.969059
std	1.429470	0.503846	5.097809	0.669196	700.752504
min	10.100000	0.900000	2.000000	0.690000	174.000000
25%	14.000000	2.000000	4.000000	1.500000	609.000000
50%	15.600000	2.500000	8.000000	2.040000	989.000000
75%	15.600000	2.700000	8.000000	2.310000	1496.500000
max	18.400000	3.600000	64.000000	4.700000	6099.000000

Add Code CellAdd Markdown Cell

51694

[15]

```
1 # Определим уникальные значения для целевого признака
2 data['Price (Euro)'].unique()
```

Executed at 2025.02.20 12:51:11 in 12ms

1029. , 2226. , 1312.49, 196. , 1513. , 523.63, 1895. ,
2050.38, 278. , 752. , 616. , 3949.4 , 784. , 2171.72,
2440. , 1142.8 , 2296.95, 1009.9 , 2339. , 339. , 2250.68,
478.89, 1492.8 , 788.49, 2041. , 1769. , 476.99, 1390. ,
679. , 1305. , 2153.37, 1637. , 831. , 895.01, 333. ,
909. , 691. , 1163. , 1327. , 368. , 2150. , 272. ,
2048.9 , 2680. , 1948.99, 174. , 1272. , 1476.11, 1713.37,
1477. , 521.86, 3499. , 469.01, 1598. , 478. , 2198.19,
737. , 597.57, 361.8 , 1600. , 875. , 2325. , 573. ,
1813. , 324. , 1072. , 443.99, 490. , 895. , 833.01,
729. , 289. , 549.99, 805.99, 720.32, 638. , 764. ])

Целевой признак является дискретной непрерывный(?) и содержит только значения 174 и 6099.

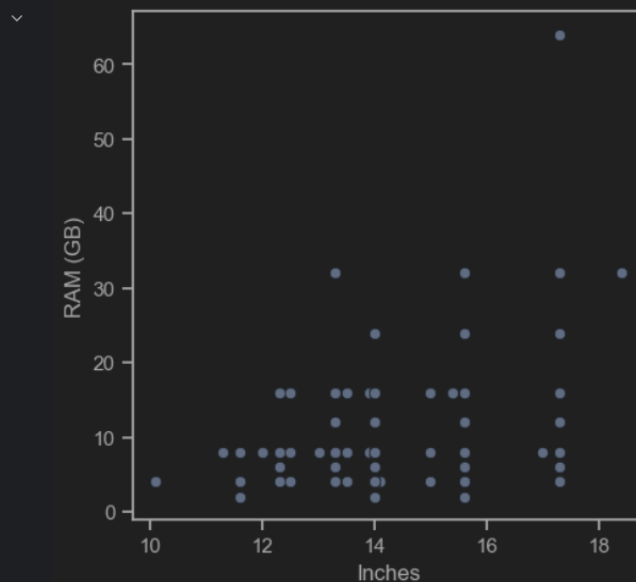
## Диаграмма рассеяния

Позволяет построить распределение двух колонок данных и визуально обнаружить наличие зависимости. Не предполагается, что значения упорядочены (например, по времени).

[36]

```
1
2 # Построение графика
3 fig, ax = plt.subplots(figsize=(5,5))
4 sns.scatterplot(ax=ax, x='Inches', y='RAM (GB)', data=data)
5
6 # Отображение графика
7 plt.show()
8
```

Executed at 2025.02.20 13:07:23 in 111ms



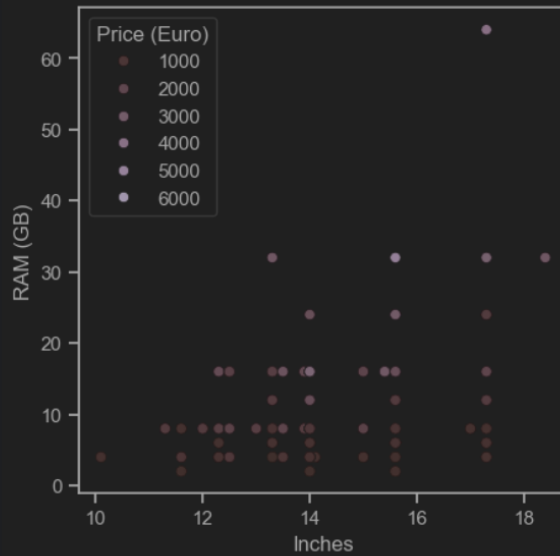
Можно видеть что между полями Inches и RAM присутствует слабая (хаотичная) зависимость.

Посмотрим насколько на эту зависимость влияет целевой признак.

5 1 694 ^ v

```
[41] 1 fig, ax = plt.subplots(figsize=(5,5))
      2 sns.scatterplot(ax=ax, x='Inches', y='RAM (GB)', hue='Price (Euro)', data=data)
      3 plt.show()
      4
```

Executed at 2025.02.20 13:11:03 in 214ms

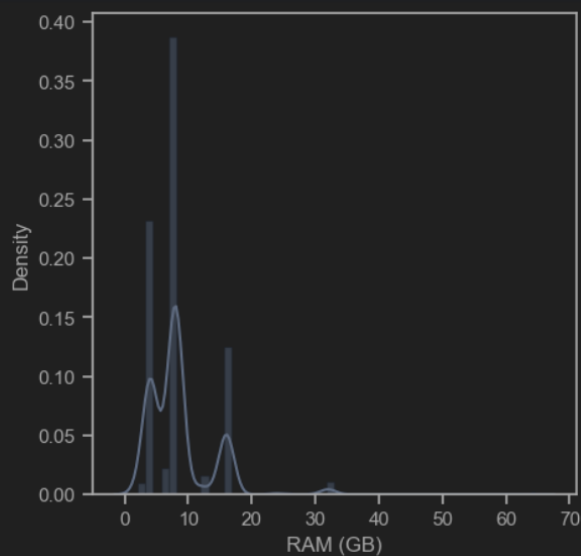


5 1 694 ^ v

```
[44] 1 fig, ax = plt.subplots(figsize=(5,5))
      2 sns.distplot(data['RAM (GB)'])
      3 plt.show()
```

Executed at 2025.02.20 13:13:16 in 413ms

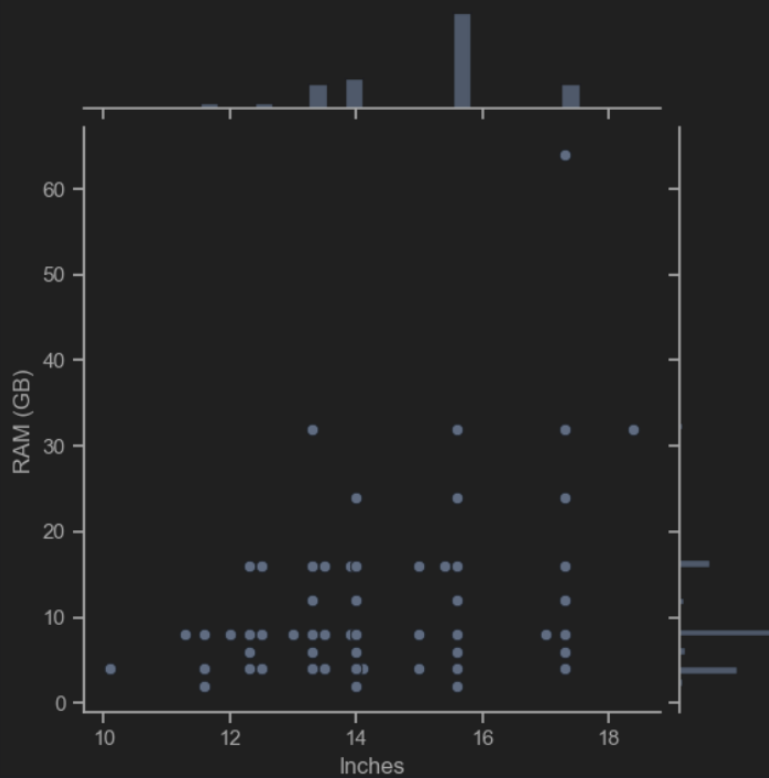
> C:\Users\kayat\AppData\Local\Temp\ipykernel\_3360\2561556947.py:2: UserWarning: \n\n'distplot' is a deprec...



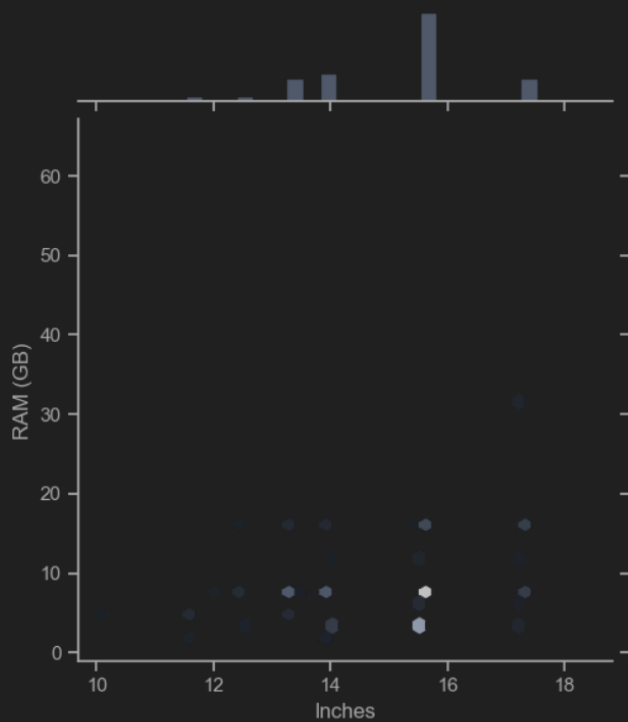
## Jointplot

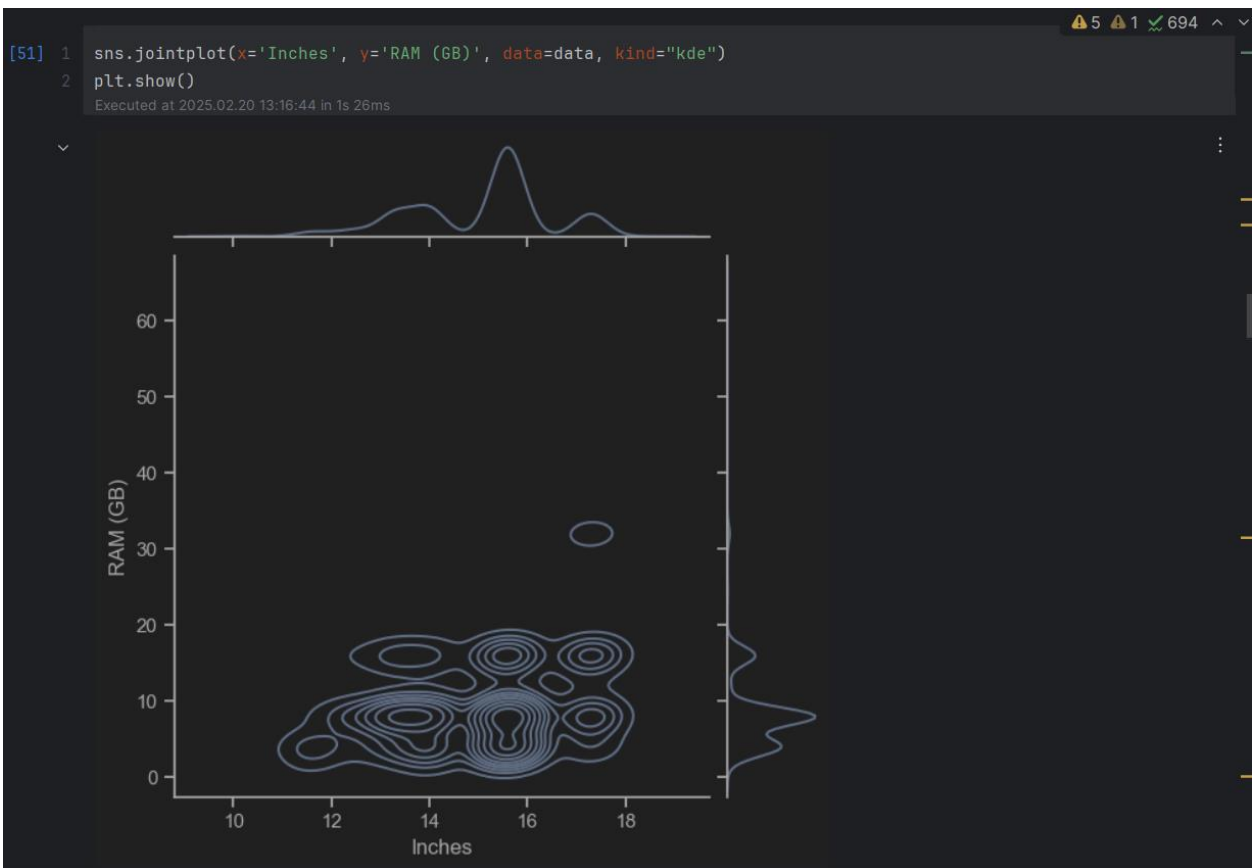
Комбинация гистограмм и диаграмм рассеивания.

```
[47] 1 sns.jointplot(x='Inches', y='RAM (GB)', data=data)
      2 plt.show()
      Executed at 2025.02.20 13:15:40 in 332ms
```



```
[48] 1 sns.jointplot(x='Inches', y='RAM (GB)', data=data, kind="hex")
      2 plt.show()
      Executed at 2025.02.20 13:16:11 in 375ms
```



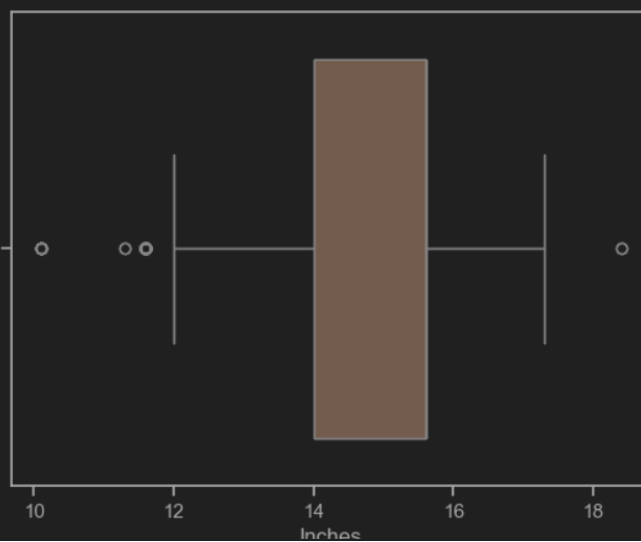


## Ящик с усами

Отображает одномерное распределение вероятности.

```
57] 1 sns.boxplot(x=data['Inches'])
      2 plt.show()
```

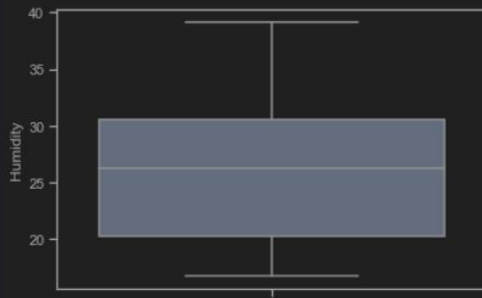
Executed at 2025.02.20 13:20:35 in 71ms





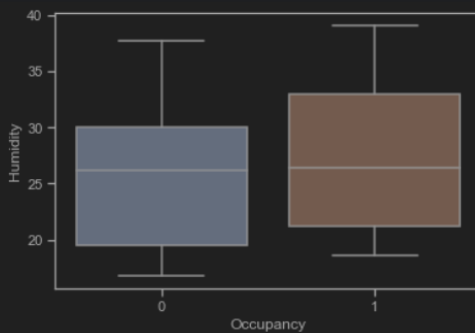
```
[20] 1 # По вертикали
      2 sns.boxplot(y=data['Humidity'])
```

<AxesSubplot:ylabel='Humidity'>



```
[21] 1 # Распределение параметра Humidity сгруппированные по Occupancy.
      2 sns.boxplot(x='Occupancy', y='Humidity', data=data)
```

<AxesSubplot:xlabel='Occupancy', ylabel='Humidity'>

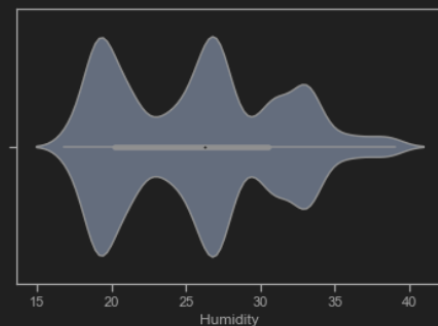


## Violin plot

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности - [https://en.wikipedia.org/wiki/Kernel\\_density\\_estimation](https://en.wikipedia.org/wiki/Kernel_density_estimation)

```
[22] 1 sns.violinplot(x=data['Humidity'])
```

<AxesSubplot:xlabel='Humidity'>

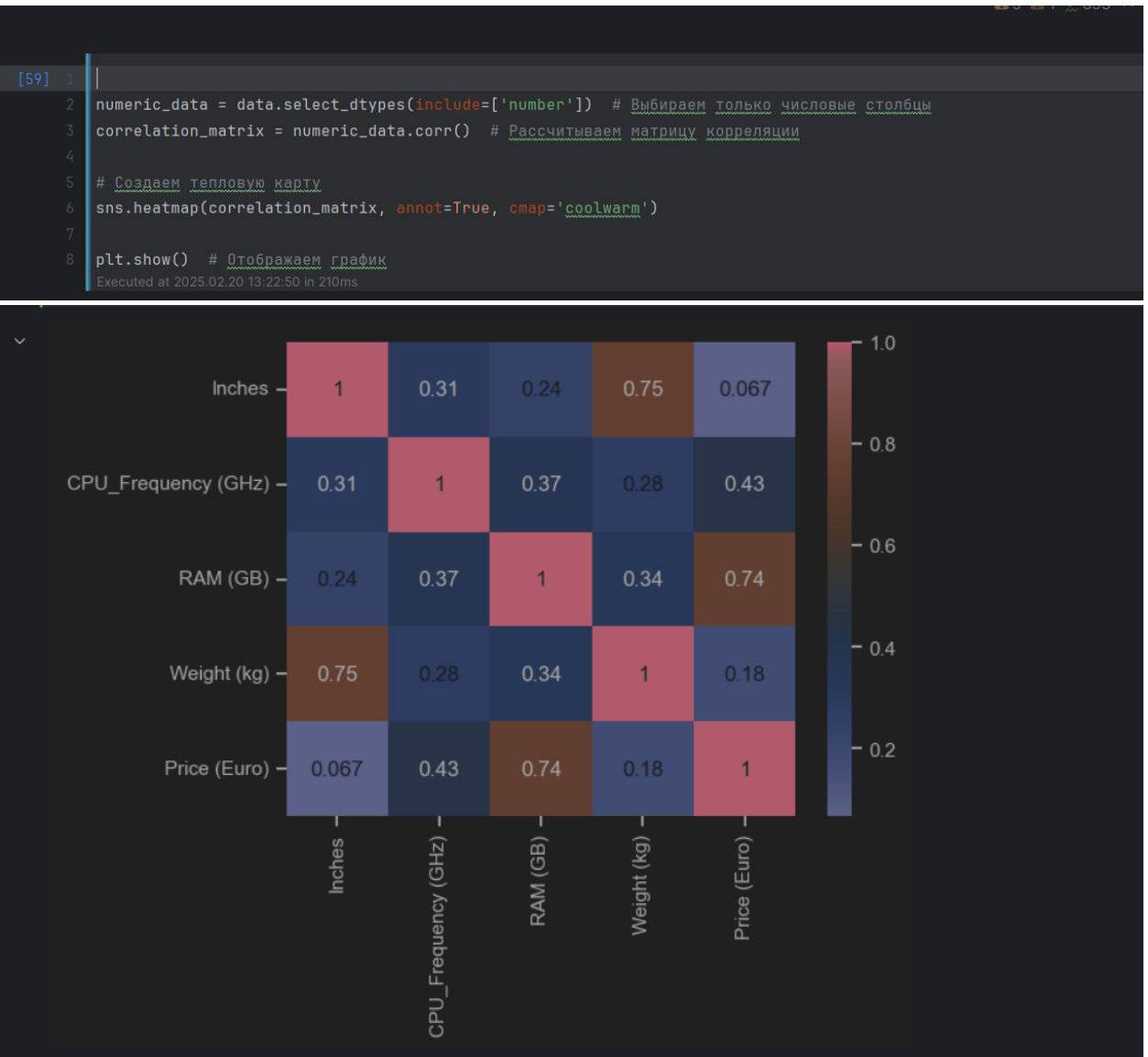


В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

Для визуализации корреляционной матрицы будем использовать "тепловую карту" heatmap которая показывает степень корреляции различными цветами.

Используем метод heatmap библиотеки seaborn -

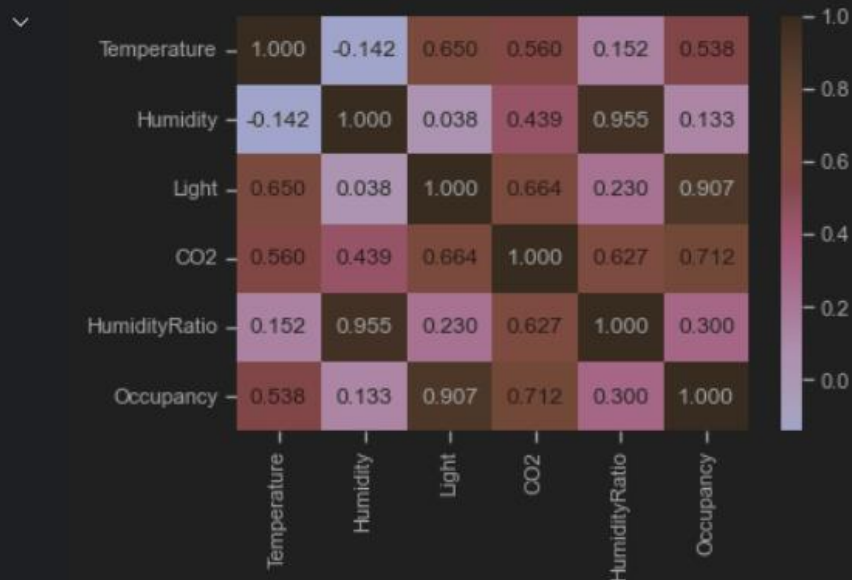
<https://seaborn.pydata.org/generated/seaborn.heatmap.html>



[Add Code Cell](#)[Add Markdown Cell](#)

```
[31] 1 # Вывод значений в ячейках
      2 sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

<AxesSubplot:>



```
[32] 1 # Изменение цветовой гаммы
      2 sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

<AxesSubplot:>

