Московский государственный технический университет им. Н. Э. Баумана

Курс «Технологии машинного обучения»

Отчёт по лабораторной работе №1

Выполнил:
Флоринский В. А.
группа ИУ5-64Б

Проверил:
Гапанюк Ю.Е.

Дата: 07.04.25

Дата:

Подпись:

Подпись:

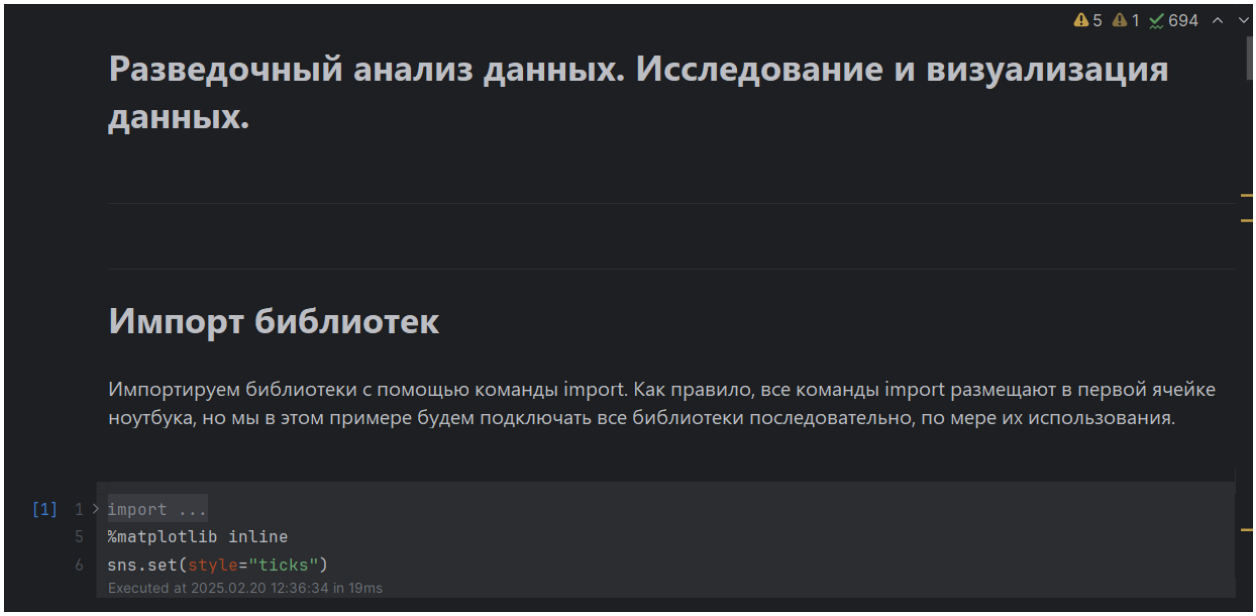
Москва, 2025 г.

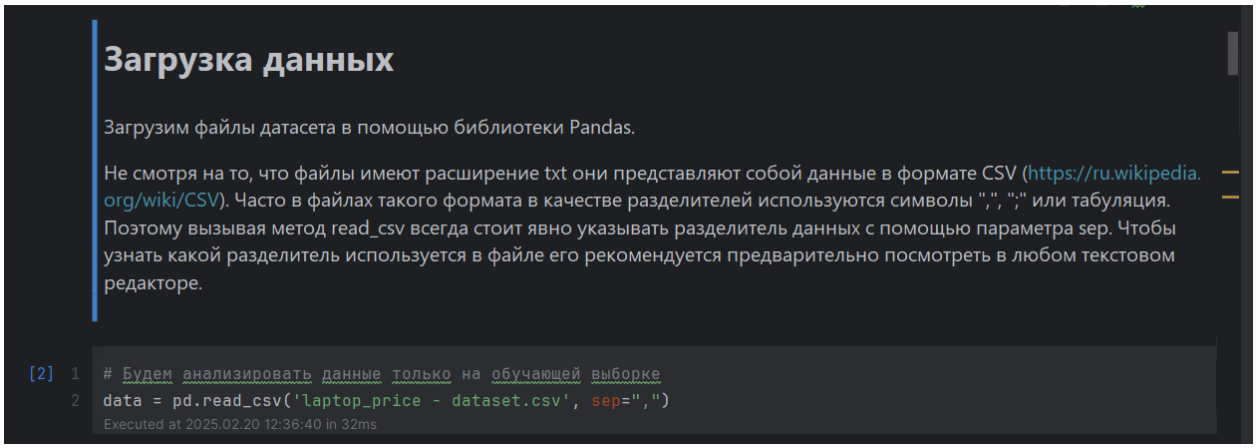
Цель лабораторной работы: изучение различных методов визуализация данных.

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.





2) Основные характеристики датасета

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

5 rows | 5 rows × 15 columns | Static Output

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

```
(1275, 15)
```

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

```
Всего строк: 1275
```

5 1 694

[6]
```
# Список колонок
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]
```
# Список колонок с типами данных
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]
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in data.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = data[data[col].isnull()].shape[0]
    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]
# Основные статистические характеристки набора данных
data.describe()
```

Executed at 2025.02.20 12:40:27 in 16ms

8 rows | 8 rows × 5 columns | Static Output

	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 Cell | Add Markdown Cell

```
[15]
# Определим уникальные значения для целевого признака
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 непрервный(?) и содержит только значения 174 и 6099.

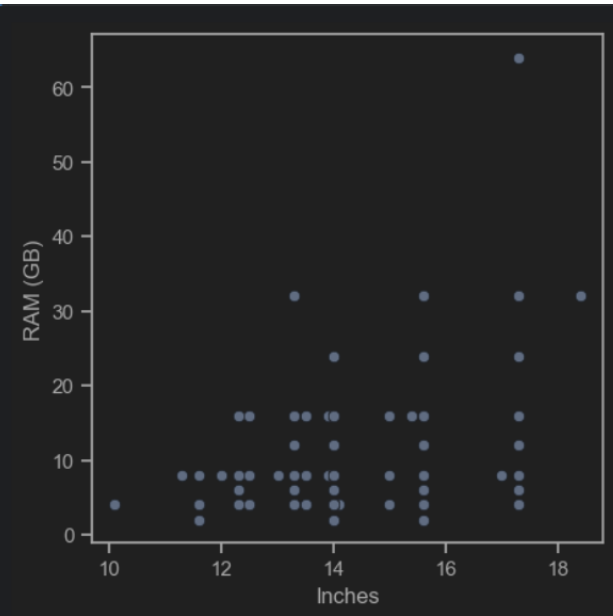
Диаграмма рассеяния

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

[36]
```
# Построение графика
fig, ax = plt.subplots(figsize=(5,5))
sns.scatterplot(ax=ax, x='Inches', y='RAM (GB)', data=data)

# Отображение графика
plt.show()
```

Executed at 2025.02.20 13:07:23 in 111ms



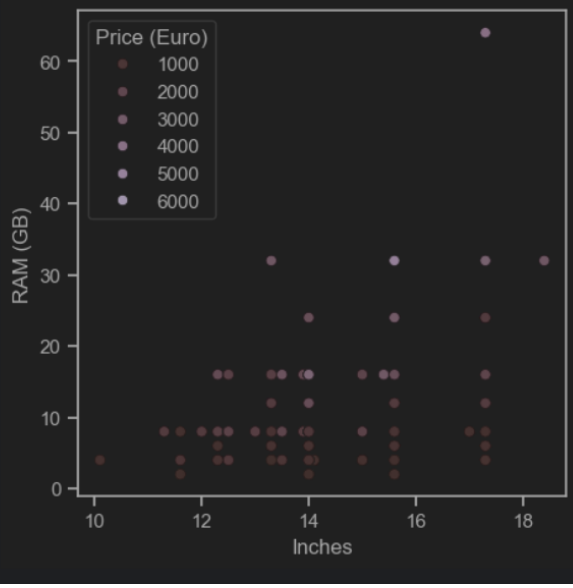
Можно видеть что между полями Inches и RAM пристутствует слабая (хаотичная) зависимость.

Посмотрим насколько на эту зависимость влияет целевой признак.

5 1 694

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

Executed at 2025.02.20 13:11:03 in 214ms

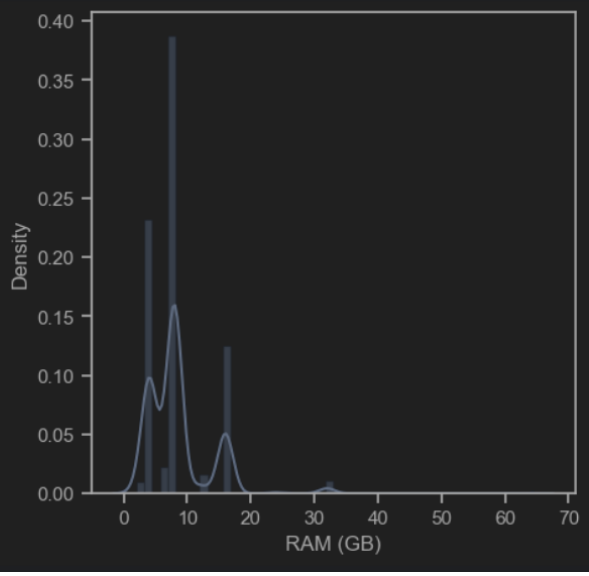


5 1 694

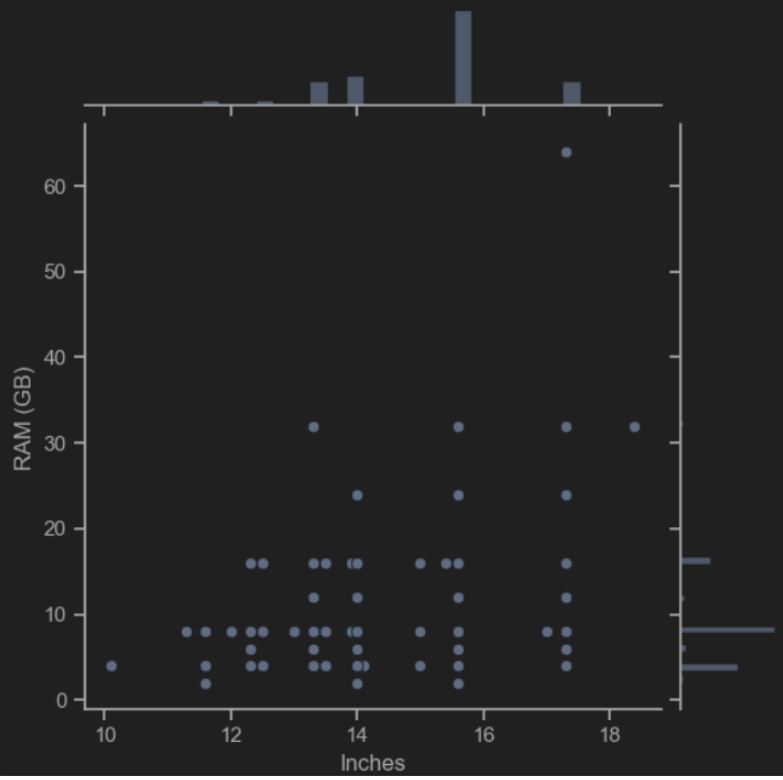
[44]
```
fig, ax = plt.subplots(figsize=(5,5))
sns.distplot(data['RAM (GB)'])
plt.show()
```

Executed at 2025.02.20 13:13:16 in 413ms

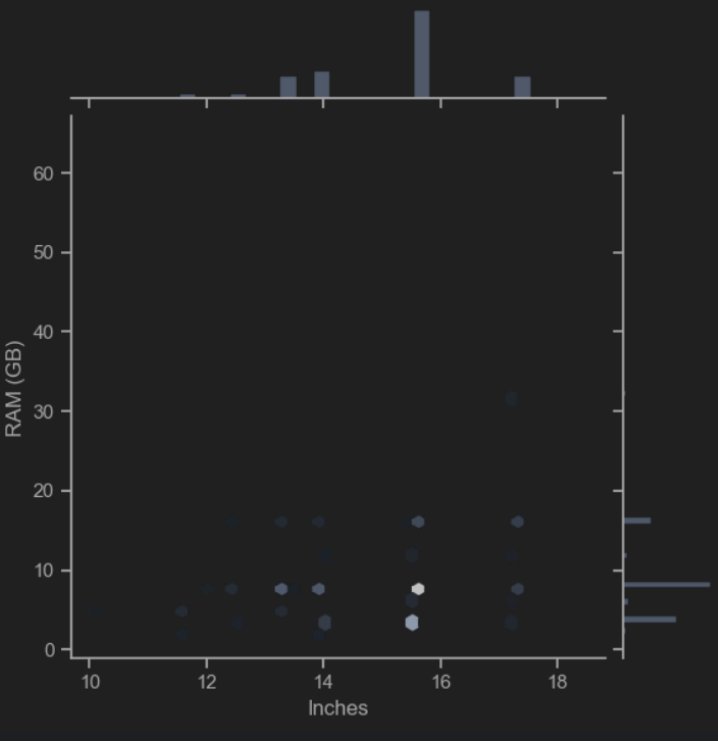
```
C:\Users\kayat\AppData\Local\Temp\ipykernel_3360\2561556947.py:2: UserWarning: \n\n`distplot` is a deprec...
```



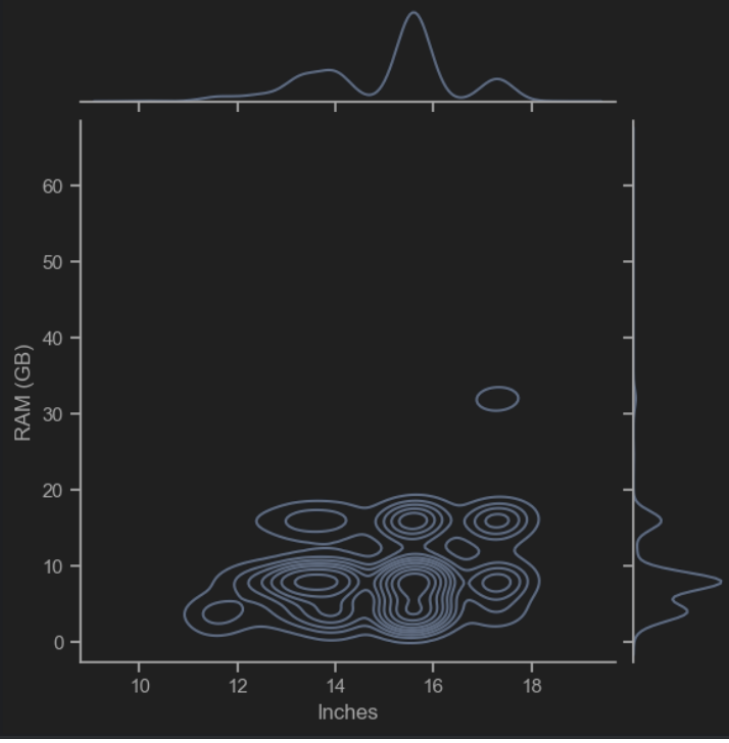
[47]
```
sns.jointplot(x='Inches', y='RAM (GB)', data=data)
plt.show()
```
Executed at 2025.02.20 13:15:40 in 332ms



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



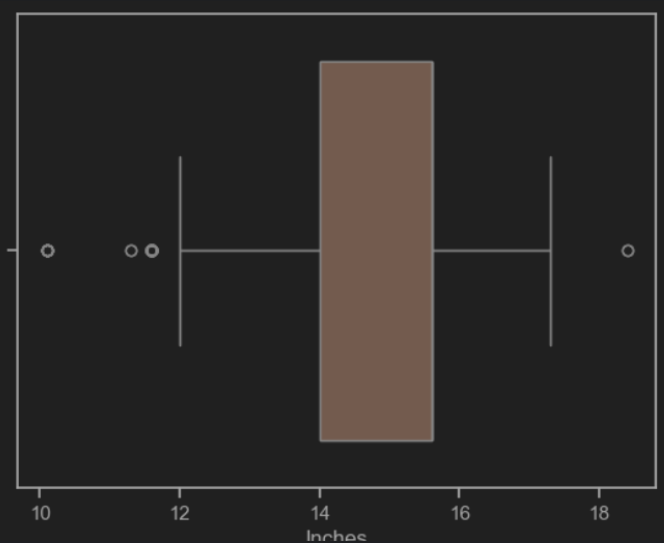
```
[51] 1 sns.jointplot(x='Inches', y='RAM (GB)', data=data, kind="kde")
     2 plt.show()
```
Executed at 2025.02.20 13:16:44 in 1s 26ms



Ящик с усами

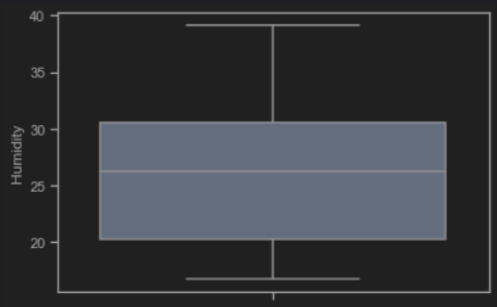
Отображает одномерное распределение вероятности.

```
[57] 1 sns.boxplot(x=data['Inches'])
     2 plt.show()
```
Executed at 2025.02.20 13:20:35 in 71ms



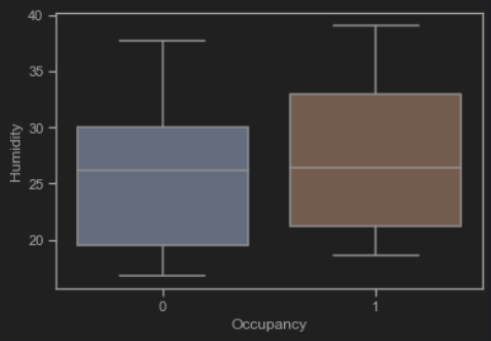
[20]
```
# По вертикали
sns.boxplot(y=data['Humidity'])
```

```
<AxesSubplot:ylabel='Humidity'>
```



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

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

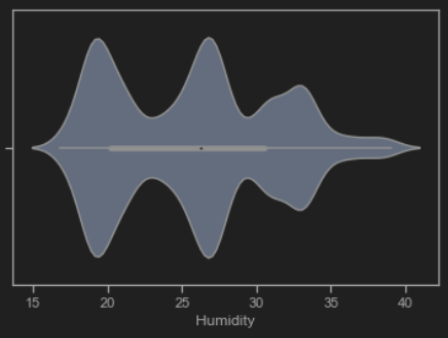


Violin plot

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности - https://en.wikipedia.org/wiki/Kernel_density_estimation

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

```
<AxesSubplot:xlabel='Humidity'>
```

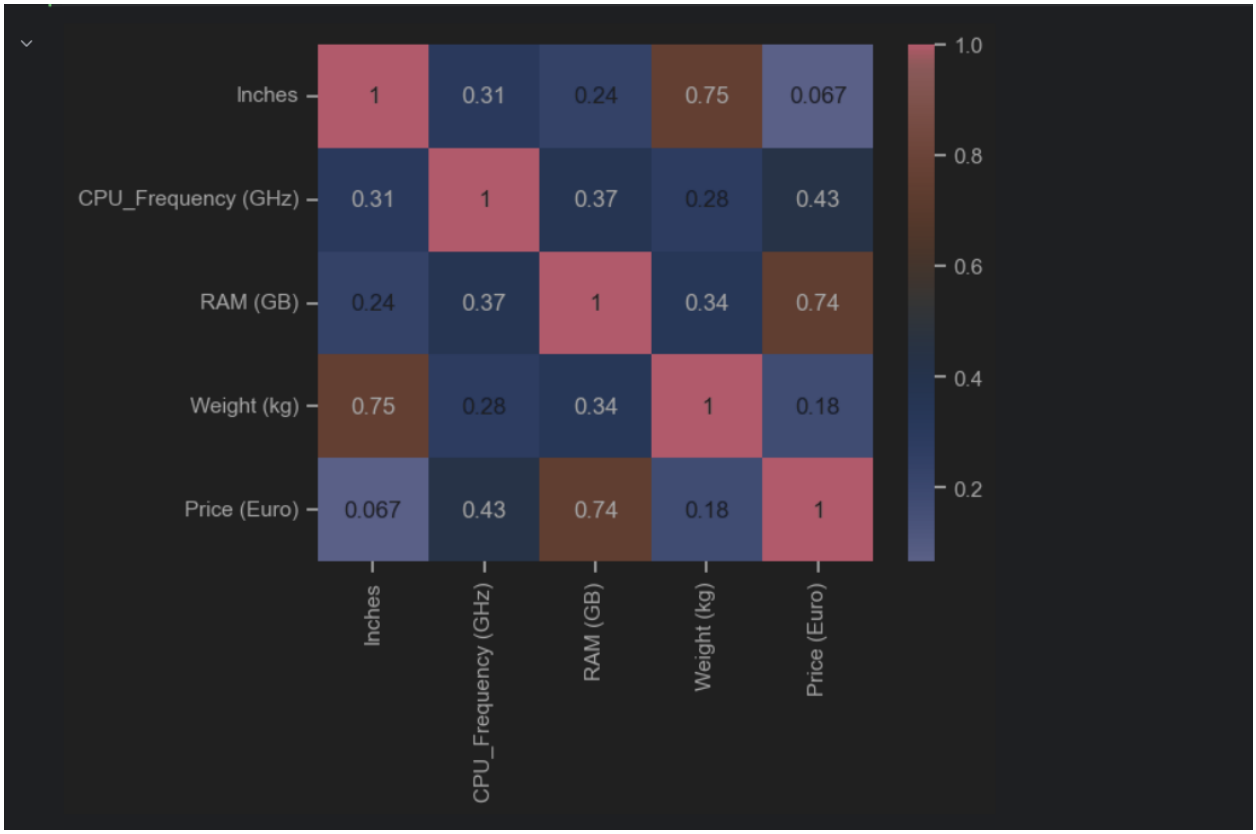


```
[59]
numeric_data = data.select_dtypes(include=['number'])  # Выбираем только числовые столбцы
correlation_matrix = numeric_data.corr()  # Рассчитываем матрицу корреляции

# Создаем тепловую карту
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')

plt.show()  # Отображаем график
```

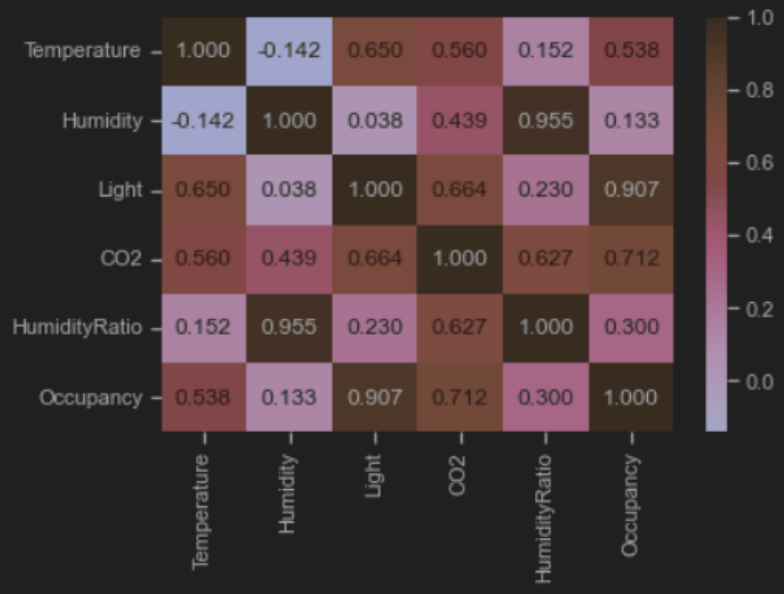
Executed at 2025.02.20 13:22:50 in 210ms



Add Code Cell | Add Markdown Cell

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

```
<AxesSubplot:>
```



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

```
<AxesSubplot:>
```