# **Разведочный анализ данных. Исследование и визуализация данных.**

## **1) Текстовое описание набора данных**

В качестве набора данных мы будем использовать набор данных c изображениями рукописных цифр.
<https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits>
Набор взят из scikit-learn из раздела Toy-datasets.
Датасет содержит следующие значения:
data - необходимые для предсказания данные в массиве NumPy
target - целевые данные в массиве NumPy - номер класса картинки (т.е. к какой цифре от 0 до 9 относится)
frame - кадр: DataFrame формы (1797, 65)
feature\_names - имена столбцов набора данных
target\_names - имена целевых классов
images - необработанные данные изображения
DESCR - описание датасета

### **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.

import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
%matplotlib inline   
sns.set(style="ticks")  
import sklearn  
from sklearn import datasets

### **Загрузка данных**

Загрузим данные из Scikit-learn и преобразуем их в Pandas Dataframe

from sklearn.datasets import load\_digits  
digits = load\_digits()  
for x in digits:  
 print(x)

data  
target  
frame  
feature\_names  
target\_names  
images  
DESCR

data = pd.DataFrame(data= np.c\_[digits['data'], digits['target']],  
 columns= digits['feature\_names'] + ['target'])

## **2) Основные характеристики датасета**

data

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
0 0.0 0.0 5.0 13.0 9.0 1.0   
1 0.0 0.0 0.0 12.0 13.0 5.0   
2 0.0 0.0 0.0 4.0 15.0 12.0   
3 0.0 0.0 7.0 15.0 13.0 1.0   
4 0.0 0.0 0.0 1.0 11.0 0.0   
... ... ... ... ... ... ...   
1792 0.0 0.0 4.0 10.0 13.0 6.0   
1793 0.0 0.0 6.0 16.0 13.0 11.0   
1794 0.0 0.0 1.0 11.0 15.0 1.0   
1795 0.0 0.0 2.0 10.0 7.0 0.0   
1796 0.0 0.0 10.0 14.0 8.0 1.0   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 pixel\_7\_0 \  
0 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1 0.0 0.0 0.0 0.0 ... 0.0 0.0   
2 0.0 0.0 0.0 0.0 ... 0.0 0.0   
3 0.0 0.0 0.0 8.0 ... 0.0 0.0   
4 0.0 0.0 0.0 0.0 ... 0.0 0.0   
... ... ... ... ... ... ... ...   
1792 0.0 0.0 0.0 1.0 ... 0.0 0.0   
1793 1.0 0.0 0.0 0.0 ... 0.0 0.0   
1794 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1795 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1796 0.0 0.0 0.0 2.0 ... 0.0 0.0   
  
 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 pixel\_7\_6 \  
0 0.0 6.0 13.0 10.0 0.0 0.0   
1 0.0 0.0 11.0 16.0 10.0 0.0   
2 0.0 0.0 3.0 11.0 16.0 9.0   
3 0.0 7.0 13.0 13.0 9.0 0.0   
4 0.0 0.0 2.0 16.0 4.0 0.0   
... ... ... ... ... ... ...   
1792 0.0 2.0 14.0 15.0 9.0 0.0   
1793 0.0 6.0 16.0 14.0 6.0 0.0   
1794 0.0 2.0 9.0 13.0 6.0 0.0   
1795 0.0 5.0 12.0 16.0 12.0 0.0   
1796 1.0 8.0 12.0 14.0 12.0 1.0   
  
 pixel\_7\_7 target   
0 0.0 0.0   
1 0.0 1.0   
2 0.0 2.0   
3 0.0 3.0   
4 0.0 4.0   
... ... ...   
1792 0.0 9.0   
1793 0.0 0.0   
1794 0.0 8.0   
1795 0.0 9.0   
1796 0.0 8.0   
  
[1797 rows x 65 columns]

# Первые 5 строк датасета - первые 5 картинок в пикселях  
data.head()

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
0 0.0 0.0 5.0 13.0 9.0 1.0   
1 0.0 0.0 0.0 12.0 13.0 5.0   
2 0.0 0.0 0.0 4.0 15.0 12.0   
3 0.0 0.0 7.0 15.0 13.0 1.0   
4 0.0 0.0 0.0 1.0 11.0 0.0   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 pixel\_7\_0 \  
0 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1 0.0 0.0 0.0 0.0 ... 0.0 0.0   
2 0.0 0.0 0.0 0.0 ... 0.0 0.0   
3 0.0 0.0 0.0 8.0 ... 0.0 0.0   
4 0.0 0.0 0.0 0.0 ... 0.0 0.0   
  
 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 pixel\_7\_6 \  
0 0.0 6.0 13.0 10.0 0.0 0.0   
1 0.0 0.0 11.0 16.0 10.0 0.0   
2 0.0 0.0 3.0 11.0 16.0 9.0   
3 0.0 7.0 13.0 13.0 9.0 0.0   
4 0.0 0.0 2.0 16.0 4.0 0.0   
  
 pixel\_7\_7 target   
0 0.0 0.0   
1 0.0 1.0   
2 0.0 2.0   
3 0.0 3.0   
4 0.0 4.0   
  
[5 rows x 65 columns]

# Размер датасета - 1797 строк, 65 колонок (картинка 8х8 и target)  
data.shape

(1797, 65)

total\_count = data.shape[0]  
print('Всего строк: {}'.format(total\_count))

Всего строк: 1797

# Список колонок  
data.columns

Index(['pixel\_0\_0', 'pixel\_0\_1', 'pixel\_0\_2', 'pixel\_0\_3', 'pixel\_0\_4',  
 'pixel\_0\_5', 'pixel\_0\_6', 'pixel\_0\_7', 'pixel\_1\_0', 'pixel\_1\_1',  
 'pixel\_1\_2', 'pixel\_1\_3', 'pixel\_1\_4', 'pixel\_1\_5', 'pixel\_1\_6',  
 'pixel\_1\_7', 'pixel\_2\_0', 'pixel\_2\_1', 'pixel\_2\_2', 'pixel\_2\_3',  
 'pixel\_2\_4', 'pixel\_2\_5', 'pixel\_2\_6', 'pixel\_2\_7', 'pixel\_3\_0',  
 'pixel\_3\_1', 'pixel\_3\_2', 'pixel\_3\_3', 'pixel\_3\_4', 'pixel\_3\_5',  
 'pixel\_3\_6', 'pixel\_3\_7', 'pixel\_4\_0', 'pixel\_4\_1', 'pixel\_4\_2',  
 'pixel\_4\_3', 'pixel\_4\_4', 'pixel\_4\_5', 'pixel\_4\_6', 'pixel\_4\_7',  
 'pixel\_5\_0', 'pixel\_5\_1', 'pixel\_5\_2', 'pixel\_5\_3', 'pixel\_5\_4',  
 'pixel\_5\_5', 'pixel\_5\_6', 'pixel\_5\_7', 'pixel\_6\_0', 'pixel\_6\_1',  
 'pixel\_6\_2', 'pixel\_6\_3', 'pixel\_6\_4', 'pixel\_6\_5', 'pixel\_6\_6',  
 'pixel\_6\_7', 'pixel\_7\_0', 'pixel\_7\_1', 'pixel\_7\_2', 'pixel\_7\_3',  
 'pixel\_7\_4', 'pixel\_7\_5', 'pixel\_7\_6', 'pixel\_7\_7', 'target'],  
 dtype='object')

# Список колонок с типами данных  
data.dtypes

pixel\_0\_0 float64  
pixel\_0\_1 float64  
pixel\_0\_2 float64  
pixel\_0\_3 float64  
pixel\_0\_4 float64  
 ...   
pixel\_7\_4 float64  
pixel\_7\_5 float64  
pixel\_7\_6 float64  
pixel\_7\_7 float64  
target float64  
Length: 65, dtype: object

# 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))

pixel\_0\_0 - 0  
pixel\_0\_1 - 0  
pixel\_0\_2 - 0  
pixel\_0\_3 - 0  
pixel\_0\_4 - 0  
pixel\_0\_5 - 0  
pixel\_0\_6 - 0  
pixel\_0\_7 - 0  
pixel\_1\_0 - 0  
pixel\_1\_1 - 0  
pixel\_1\_2 - 0  
pixel\_1\_3 - 0  
pixel\_1\_4 - 0  
pixel\_1\_5 - 0  
pixel\_1\_6 - 0  
pixel\_1\_7 - 0  
pixel\_2\_0 - 0  
pixel\_2\_1 - 0  
pixel\_2\_2 - 0  
pixel\_2\_3 - 0  
pixel\_2\_4 - 0  
pixel\_2\_5 - 0  
pixel\_2\_6 - 0  
pixel\_2\_7 - 0  
pixel\_3\_0 - 0  
pixel\_3\_1 - 0  
pixel\_3\_2 - 0  
pixel\_3\_3 - 0  
pixel\_3\_4 - 0  
pixel\_3\_5 - 0  
pixel\_3\_6 - 0  
pixel\_3\_7 - 0  
pixel\_4\_0 - 0  
pixel\_4\_1 - 0  
pixel\_4\_2 - 0  
pixel\_4\_3 - 0  
pixel\_4\_4 - 0  
pixel\_4\_5 - 0  
pixel\_4\_6 - 0  
pixel\_4\_7 - 0  
pixel\_5\_0 - 0  
pixel\_5\_1 - 0  
pixel\_5\_2 - 0  
pixel\_5\_3 - 0  
pixel\_5\_4 - 0  
pixel\_5\_5 - 0  
pixel\_5\_6 - 0  
pixel\_5\_7 - 0  
pixel\_6\_0 - 0  
pixel\_6\_1 - 0  
pixel\_6\_2 - 0  
pixel\_6\_3 - 0  
pixel\_6\_4 - 0  
pixel\_6\_5 - 0  
pixel\_6\_6 - 0  
pixel\_6\_7 - 0  
pixel\_7\_0 - 0  
pixel\_7\_1 - 0  
pixel\_7\_2 - 0  
pixel\_7\_3 - 0  
pixel\_7\_4 - 0  
pixel\_7\_5 - 0  
pixel\_7\_6 - 0  
pixel\_7\_7 - 0  
target - 0

### data.decribe()

DataFrame.count - количество не null наблюдений
DataFrame.max - максимум значений в объекте
DataFrame.мин - минимум значений в объекте
DataFrame.mean - среднее значение
DataFrame.std - стандартное отклонение наблюдений
DataFrame.select\_dtypes - подмножество DataFrame, включая/исключая столбцы на основе их dtype, в перцентилях

**Перцентиль** - это показатель, используемый в статистике, показывающий значение, ниже которого падает определенный процент наблюдений в группе наблюдений

*Пояснение:*

![изображение_2022-02-16_174912.png](data:image/png;base64;base64,)

# Основные статистические характеристки набора данных  
data.describe()

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 \  
count 1797.0 1797.000000 1797.000000 1797.000000 1797.000000   
mean 0.0 0.303840 5.204786 11.835838 11.848080   
std 0.0 0.907192 4.754826 4.248842 4.287388   
min 0.0 0.000000 0.000000 0.000000 0.000000   
25% 0.0 0.000000 1.000000 10.000000 10.000000   
50% 0.0 0.000000 4.000000 13.000000 13.000000   
75% 0.0 0.000000 9.000000 15.000000 15.000000   
max 0.0 8.000000 16.000000 16.000000 16.000000   
  
 pixel\_0\_5 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... \  
count 1797.000000 1797.000000 1797.000000 1797.000000 1797.000000 ...   
mean 5.781859 1.362270 0.129661 0.005565 1.993879 ...   
std 5.666418 3.325775 1.037383 0.094222 3.196160 ...   
min 0.000000 0.000000 0.000000 0.000000 0.000000 ...   
25% 0.000000 0.000000 0.000000 0.000000 0.000000 ...   
50% 4.000000 0.000000 0.000000 0.000000 0.000000 ...   
75% 11.000000 0.000000 0.000000 0.000000 3.000000 ...   
max 16.000000 16.000000 15.000000 2.000000 16.000000 ...   
  
 pixel\_6\_7 pixel\_7\_0 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 \  
count 1797.000000 1797.000000 1797.000000 1797.000000 1797.000000   
mean 0.206455 0.000556 0.279354 5.557596 12.089037   
std 0.984401 0.023590 0.934302 5.103019 4.374694   
min 0.000000 0.000000 0.000000 0.000000 0.000000   
25% 0.000000 0.000000 0.000000 1.000000 11.000000   
50% 0.000000 0.000000 0.000000 4.000000 13.000000   
75% 0.000000 0.000000 0.000000 10.000000 16.000000   
max 13.000000 1.000000 9.000000 16.000000 16.000000   
  
 pixel\_7\_4 pixel\_7\_5 pixel\_7\_6 pixel\_7\_7 target   
count 1797.000000 1797.000000 1797.000000 1797.000000 1797.000000   
mean 11.809126 6.764051 2.067891 0.364496 4.490818   
std 4.933947 5.900623 4.090548 1.860122 2.865304   
min 0.000000 0.000000 0.000000 0.000000 0.000000   
25% 10.000000 0.000000 0.000000 0.000000 2.000000   
50% 14.000000 6.000000 0.000000 0.000000 4.000000   
75% 16.000000 12.000000 2.000000 0.000000 7.000000   
max 16.000000 16.000000 16.000000 16.000000 9.000000   
  
[8 rows x 65 columns]

# Определим уникальные значения для целевого признака  
data['target'].unique()

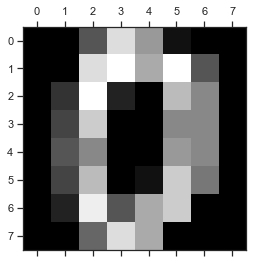
array([0., 1., 2., 3., 4., 5., 6., 7., 8., 9.])

Целевой признак содержит все уникальные классы картинок - цифры от 0 до 9

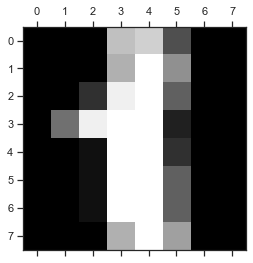
Выведем первые 5 картинок датасета

for i in range(5):  
 plt.gray()  
 plt.matshow(digits.images[i])  
 plt.show()

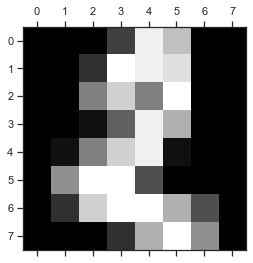
<Figure size 432x288 with 0 Axes>



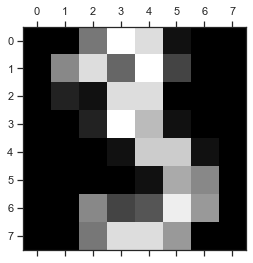
<Figure size 432x288 with 0 Axes>



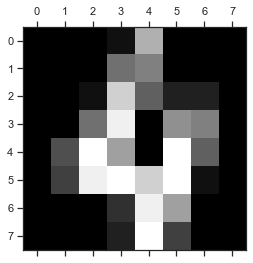
<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>



<Figure size 432x288 with 0 Axes>

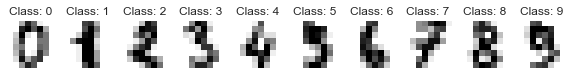


## **3) Визуальное исследование датасета**

Для визуального исследования могут быть использованы различные виды диаграмм, мы построим только некоторые варианты диаграмм, которые используются достаточно часто.

### **Imshow**

fig, axes = plt.subplots(nrows=1, ncols=10, figsize=(10, 3))  
for ax, image, label in zip(axes, digits.images, digits.target):  
 ax.set\_axis\_off()  
 ax.imshow(image, cmap=plt.cm.gray\_r, interpolation="nearest")  
 ax.set\_title("Class: %i" % label)



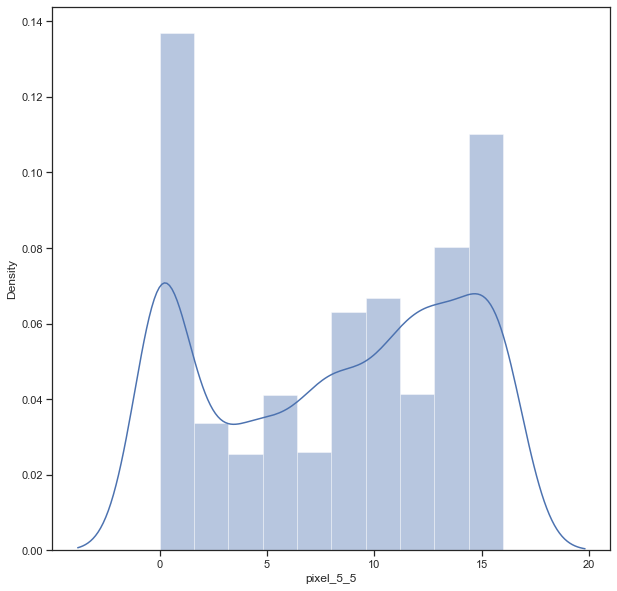
### **Гистограмма**

Позволяет оценить плотность вероятности распределения данных. Посторим диаграмму для какого-либо пикселя из нашего датасета и посмотрим, какие значения он чаще всего принимает.

fig, ax = plt.subplots(figsize=(10,10))   
sns.distplot(data['pixel\_5\_5'])

C:\Users\Pocht\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='pixel\_5\_5', ylabel='Density'>

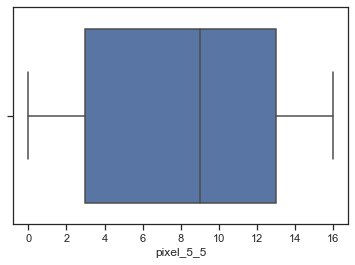


### **Ящик с усами**

Отображает одномерное распределение вероятности.

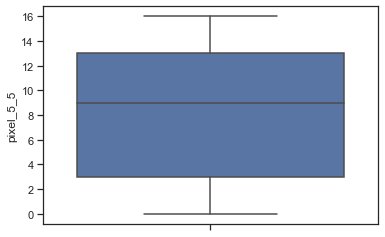
sns.boxplot(x=data['pixel\_5\_5'])

<AxesSubplot:xlabel='pixel\_5\_5'>



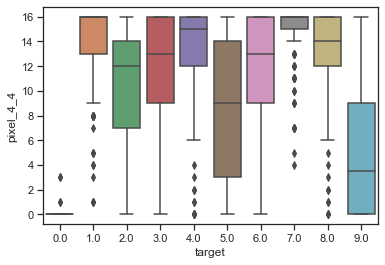
# По вертикали  
sns.boxplot(y=data['pixel\_5\_5'])

<AxesSubplot:ylabel='pixel\_5\_5'>



# Распределение параметра сгруппированное по target.  
sns.boxplot(x='target', y='pixel\_4\_4', data=data)

<AxesSubplot:xlabel='target', ylabel='pixel\_4\_4'>



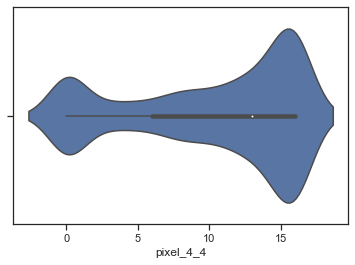
Взяли пиксель 4\_4 посередине и, как видим в цифре 0 он практически не встречается

### **Violin plot**

Похоже на предыдущую диаграмму, но по краям отображаются распределения плотности

sns.violinplot(x=data['pixel\_4\_4'])

<AxesSubplot:xlabel='pixel\_4\_4'>

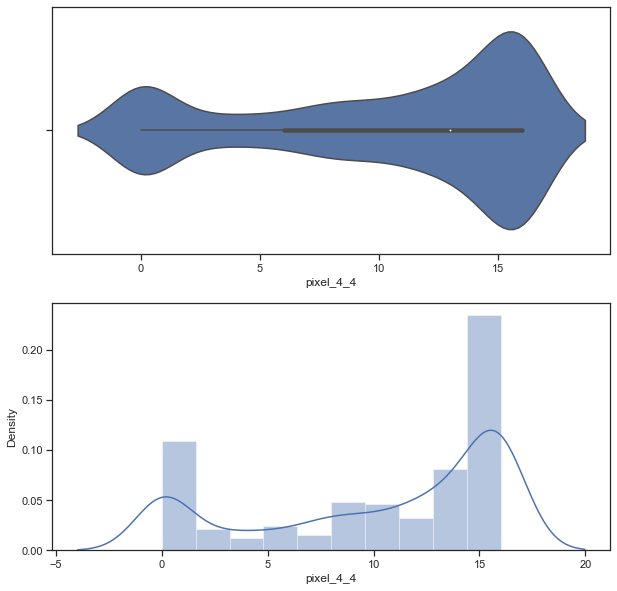


### **Subplots**

fig, ax = plt.subplots(2, 1, figsize=(10,10))  
sns.violinplot(ax=ax[0], x=data['pixel\_4\_4'])  
sns.distplot(data['pixel\_4\_4'], ax=ax[1])

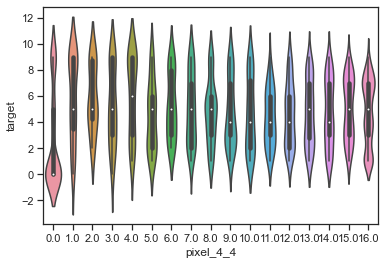
C:\Users\Pocht\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
 warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='pixel\_4\_4', ylabel='Density'>



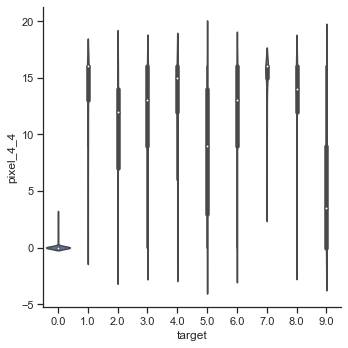
# Распределение параметра сгруппированное по target.  
sns.violinplot(x='pixel\_4\_4', y='target', data=data)

<AxesSubplot:xlabel='pixel\_4\_4', ylabel='target'>



sns.catplot(y='pixel\_4\_4', x='target', data=data, kind="violin", split=True)

<seaborn.axisgrid.FacetGrid at 0x22aed2b7eb0>



## **4) Информация о корреляции признаков**

Проверка корреляции признаков позволяет решить две задачи:

Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (в нашем примере это колонка "target"). Именно эти признаки будут наиболее информативными для моделей машинного обучения. Признаки, которые слабо коррелируют с целевым признаком, можно попробовать исключить из построения модели, иногда это повышает качество модели. Нужно отметить, что некоторые алгоритмы машинного обучения автоматически определяют ценность того или иного признака для построения модели.
Понять какие нецелевые признаки линейно зависимы между собой. Линейно зависимые признаки, как правило, очень плохо влияют на качество моделей. Поэтому если несколько признаков линейно зависимы, то для построения модели из них выбирают какой-то один признак.

data.corr()

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 NaN 1.000000 0.556618 0.207814 -0.018761 0.060776   
pixel\_0\_2 NaN 0.556618 1.000000 0.560180 -0.084235 0.043569   
pixel\_0\_3 NaN 0.207814 0.560180 1.000000 0.023938 -0.171377   
pixel\_0\_4 NaN -0.018761 -0.084235 0.023938 1.000000 0.507731   
... ... ... ... ... ... ...   
pixel\_7\_4 NaN -0.102349 -0.134754 -0.065957 -0.082125 -0.351146   
pixel\_7\_5 NaN -0.029870 -0.041183 -0.054936 -0.215809 -0.268818   
pixel\_7\_6 NaN 0.026547 0.072599 0.053437 -0.250699 -0.267659   
pixel\_7\_7 NaN -0.043889 0.082523 0.081971 -0.215349 -0.167727   
target NaN -0.051834 -0.011836 -0.011489 0.100801 0.193362   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 \  
pixel\_0\_0 NaN NaN NaN NaN ... NaN   
pixel\_0\_1 0.048388 -0.038927 0.032320 0.556372 ... -0.045342   
pixel\_0\_2 0.002841 -0.062278 0.022311 0.582259 ... -0.003804   
pixel\_0\_3 -0.115732 -0.040139 0.035663 0.328344 ... 0.075335   
pixel\_0\_4 0.127764 0.010065 0.042065 0.051657 ... -0.212220   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.383522 -0.178243 0.048996 -0.000604 ... 0.105101   
pixel\_7\_5 -0.304111 -0.141174 0.033409 0.071488 ... 0.262795   
pixel\_7\_6 -0.178945 -0.063220 0.020689 0.111569 ... 0.511726   
pixel\_7\_7 -0.080309 -0.024505 -0.005226 -0.001404 ... 0.563989   
target 0.197343 0.101085 0.020813 -0.012439 ... -0.099312   
  
 pixel\_7\_0 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 -0.007905 0.855610 0.555727 0.147646 -0.102349 -0.029870   
pixel\_0\_2 -0.025837 0.515276 0.937623 0.499840 -0.134754 -0.041183   
pixel\_0\_3 -0.049085 0.175804 0.560326 0.767945 -0.065957 -0.054936   
pixel\_0\_4 0.017352 -0.047223 -0.020176 -0.008867 -0.082125 -0.215809   
... ... ... ... ... ... ...   
pixel\_7\_4 0.005697 -0.117908 -0.207957 0.058390 1.000000 0.609515   
pixel\_7\_5 -0.003056 -0.043889 -0.091839 -0.094956 0.609515 1.000000   
pixel\_7\_6 -0.011932 0.014557 0.035369 0.006849 0.243305 0.648328   
pixel\_7\_7 -0.004625 -0.047089 0.031193 0.099671 0.103015 0.261991   
target -0.020518 -0.053950 0.006367 -0.019790 -0.196083 -0.079100   
  
 pixel\_7\_6 pixel\_7\_7 target   
pixel\_0\_0 NaN NaN NaN   
pixel\_0\_1 0.026547 -0.043889 -0.051834   
pixel\_0\_2 0.072599 0.082523 -0.011836   
pixel\_0\_3 0.053437 0.081971 -0.011489   
pixel\_0\_4 -0.250699 -0.215349 0.100801   
... ... ... ...   
pixel\_7\_4 0.243305 0.103015 -0.196083   
pixel\_7\_5 0.648328 0.261991 -0.079100   
pixel\_7\_6 1.000000 0.620428 -0.114007   
pixel\_7\_7 0.620428 1.000000 -0.181929   
target -0.114007 -0.181929 1.000000   
  
[65 rows x 65 columns]

Корреляционная матрица содержит коэффициенты корреляции между всеми парами признаков.
Корреляционная матрица симметрична относительно главной диагонали. На главной диагонали расположены единицы (корреляция признака самого с собой).
На основе корреляционной матрицы можно сделать следующие выводы:
Целевой признак как-то коррелирует с пикселями 1\_6, 3\_3, 3\_4, 3\_5, 4\_3. Эти пиксели обязательно следует оставить в модели. Целевой признак отчасти коррелирует с пикселями 0\_4, 0\_5, 0\_6, 0\_7, 1\_2, 1\_3, 1\_4, 2\_3, 2\_4, 2\_5, 3\_1, 4\_1, 4\_4, 4\_5, 5\_1, 5\_2, 6\_3, 6\_4 Особо сильно коррелирующих между собой пикселей нет, и для точности изображения лучше оставить слабо кррелирующие.

Описание метода corr - <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.corr.html>

По умолчанию при построении матрицы используется коэффициент корреляции Пирсона. Возможно также построить корреляционную матрицу на основе коэффициентов корреляции Кендалла и Спирмена. На практике три метода редко дают значимые различия.

data.corr(method='pearson')

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 NaN 1.000000 0.556618 0.207814 -0.018761 0.060776   
pixel\_0\_2 NaN 0.556618 1.000000 0.560180 -0.084235 0.043569   
pixel\_0\_3 NaN 0.207814 0.560180 1.000000 0.023938 -0.171377   
pixel\_0\_4 NaN -0.018761 -0.084235 0.023938 1.000000 0.507731   
... ... ... ... ... ... ...   
pixel\_7\_4 NaN -0.102349 -0.134754 -0.065957 -0.082125 -0.351146   
pixel\_7\_5 NaN -0.029870 -0.041183 -0.054936 -0.215809 -0.268818   
pixel\_7\_6 NaN 0.026547 0.072599 0.053437 -0.250699 -0.267659   
pixel\_7\_7 NaN -0.043889 0.082523 0.081971 -0.215349 -0.167727   
target NaN -0.051834 -0.011836 -0.011489 0.100801 0.193362   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 \  
pixel\_0\_0 NaN NaN NaN NaN ... NaN   
pixel\_0\_1 0.048388 -0.038927 0.032320 0.556372 ... -0.045342   
pixel\_0\_2 0.002841 -0.062278 0.022311 0.582259 ... -0.003804   
pixel\_0\_3 -0.115732 -0.040139 0.035663 0.328344 ... 0.075335   
pixel\_0\_4 0.127764 0.010065 0.042065 0.051657 ... -0.212220   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.383522 -0.178243 0.048996 -0.000604 ... 0.105101   
pixel\_7\_5 -0.304111 -0.141174 0.033409 0.071488 ... 0.262795   
pixel\_7\_6 -0.178945 -0.063220 0.020689 0.111569 ... 0.511726   
pixel\_7\_7 -0.080309 -0.024505 -0.005226 -0.001404 ... 0.563989   
target 0.197343 0.101085 0.020813 -0.012439 ... -0.099312   
  
 pixel\_7\_0 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 -0.007905 0.855610 0.555727 0.147646 -0.102349 -0.029870   
pixel\_0\_2 -0.025837 0.515276 0.937623 0.499840 -0.134754 -0.041183   
pixel\_0\_3 -0.049085 0.175804 0.560326 0.767945 -0.065957 -0.054936   
pixel\_0\_4 0.017352 -0.047223 -0.020176 -0.008867 -0.082125 -0.215809   
... ... ... ... ... ... ...   
pixel\_7\_4 0.005697 -0.117908 -0.207957 0.058390 1.000000 0.609515   
pixel\_7\_5 -0.003056 -0.043889 -0.091839 -0.094956 0.609515 1.000000   
pixel\_7\_6 -0.011932 0.014557 0.035369 0.006849 0.243305 0.648328   
pixel\_7\_7 -0.004625 -0.047089 0.031193 0.099671 0.103015 0.261991   
target -0.020518 -0.053950 0.006367 -0.019790 -0.196083 -0.079100   
  
 pixel\_7\_6 pixel\_7\_7 target   
pixel\_0\_0 NaN NaN NaN   
pixel\_0\_1 0.026547 -0.043889 -0.051834   
pixel\_0\_2 0.072599 0.082523 -0.011836   
pixel\_0\_3 0.053437 0.081971 -0.011489   
pixel\_0\_4 -0.250699 -0.215349 0.100801   
... ... ... ...   
pixel\_7\_4 0.243305 0.103015 -0.196083   
pixel\_7\_5 0.648328 0.261991 -0.079100   
pixel\_7\_6 1.000000 0.620428 -0.114007   
pixel\_7\_7 0.620428 1.000000 -0.181929   
target -0.114007 -0.181929 1.000000   
  
[65 rows x 65 columns]

data.corr(method='kendall')

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
pixel\_0\_0 1.0 NaN NaN NaN NaN NaN   
pixel\_0\_1 NaN 1.000000 0.465882 0.256522 0.038614 0.029441   
pixel\_0\_2 NaN 0.465882 1.000000 0.506751 0.011530 0.037044   
pixel\_0\_3 NaN 0.256522 0.506751 1.000000 0.079054 -0.087038   
pixel\_0\_4 NaN 0.038614 0.011530 0.079054 1.000000 0.496506   
... ... ... ... ... ... ...   
pixel\_7\_4 NaN -0.033074 -0.066101 0.008530 -0.046014 -0.207973   
pixel\_7\_5 NaN -0.003165 -0.034704 -0.024476 -0.161454 -0.196805   
pixel\_7\_6 NaN 0.056966 0.025642 0.035779 -0.183123 -0.242094   
pixel\_7\_7 NaN 0.000990 0.091449 0.101062 -0.155410 -0.197487   
target NaN -0.040492 0.008607 -0.042824 0.080654 0.132804   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 \  
pixel\_0\_0 NaN NaN NaN NaN ... NaN   
pixel\_0\_1 0.062169 -0.048597 0.067609 0.515746 ... -0.017027   
pixel\_0\_2 0.026456 -0.053528 0.026521 0.505813 ... 0.029622   
pixel\_0\_3 -0.095483 -0.059491 0.029306 0.294348 ... 0.065269   
pixel\_0\_4 0.181453 -0.000730 0.044320 0.097564 ... -0.155221   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.300862 -0.159818 0.060314 0.020910 ... 0.114478   
pixel\_7\_5 -0.255171 -0.164188 0.034370 0.041364 ... 0.269659   
pixel\_7\_6 -0.223923 -0.105993 0.015769 0.090812 ... 0.423324   
pixel\_7\_7 -0.131793 -0.041462 0.019413 0.031035 ... 0.696051   
target 0.148738 0.112793 0.014800 0.021185 ... -0.084158   
  
 pixel\_7\_0 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 -0.009556 0.677321 0.471248 0.221563 -0.033074 -0.003165   
pixel\_0\_2 -0.026739 0.432331 0.835192 0.486278 -0.066101 -0.034704   
pixel\_0\_3 -0.030482 0.207891 0.507393 0.468383 0.008530 -0.024476   
pixel\_0\_4 0.014554 -0.012783 0.038562 0.053911 -0.046014 -0.161454   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.003947 -0.051525 -0.091247 0.083071 1.000000 0.530071   
pixel\_7\_5 -0.000315 -0.023451 -0.057295 -0.048574 0.530071 1.000000   
pixel\_7\_6 -0.015185 0.027867 0.016049 -0.015865 0.307113 0.660547   
pixel\_7\_7 -0.005940 -0.012840 0.060693 0.087270 0.095293 0.264027   
target -0.017617 -0.030432 0.014286 -0.049156 -0.083886 -0.076228   
  
 pixel\_7\_6 pixel\_7\_7 target   
pixel\_0\_0 NaN NaN NaN   
pixel\_0\_1 0.056966 0.000990 -0.040492   
pixel\_0\_2 0.025642 0.091449 0.008607   
pixel\_0\_3 0.035779 0.101062 -0.042824   
pixel\_0\_4 -0.183123 -0.155410 0.080654   
... ... ... ...   
pixel\_7\_4 0.307113 0.095293 -0.083886   
pixel\_7\_5 0.660547 0.264027 -0.076228   
pixel\_7\_6 1.000000 0.447316 -0.048620   
pixel\_7\_7 0.447316 1.000000 -0.143233   
target -0.048620 -0.143233 1.000000   
  
[65 rows x 65 columns]

data.corr(method='spearman')

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 NaN 1.000000 0.546549 0.306121 0.045618 0.034906   
pixel\_0\_2 NaN 0.546549 1.000000 0.643599 0.014205 0.049032   
pixel\_0\_3 NaN 0.306121 0.643599 1.000000 0.109956 -0.118404   
pixel\_0\_4 NaN 0.045618 0.014205 0.109956 1.000000 0.630281   
... ... ... ... ... ... ...   
pixel\_7\_4 NaN -0.038959 -0.090226 0.010499 -0.068580 -0.280185   
pixel\_7\_5 NaN -0.003872 -0.047479 -0.032816 -0.217234 -0.258933   
pixel\_7\_6 NaN 0.062895 0.033075 0.045415 -0.229098 -0.301882   
pixel\_7\_7 NaN 0.001071 0.109046 0.119529 -0.183080 -0.233633   
target NaN -0.048067 0.003901 -0.067370 0.115201 0.188733   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 \  
pixel\_0\_0 NaN NaN NaN NaN ... NaN   
pixel\_0\_1 0.067141 -0.050107 0.069335 0.574771 ... -0.017636   
pixel\_0\_2 0.032624 -0.063069 0.031093 0.628635 ... 0.035197   
pixel\_0\_3 -0.116923 -0.069967 0.034186 0.369092 ... 0.077149   
pixel\_0\_4 0.221711 -0.000879 0.051690 0.122804 ... -0.183476   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.364332 -0.185451 0.069643 0.026076 ... 0.133914   
pixel\_7\_5 -0.312460 -0.192483 0.040156 0.051320 ... 0.318113   
pixel\_7\_6 -0.253628 -0.114962 0.017001 0.106297 ... 0.461842   
pixel\_7\_7 -0.140970 -0.042270 0.019685 0.034579 ... 0.714031   
target 0.183245 0.132441 0.017299 0.027803 ... -0.098901   
  
 pixel\_7\_0 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 \  
pixel\_0\_0 NaN NaN NaN NaN NaN NaN   
pixel\_0\_1 -0.009799 0.695570 0.554887 0.262853 -0.038959 -0.003872   
pixel\_0\_2 -0.031345 0.509189 0.943023 0.616808 -0.090226 -0.047479   
pixel\_0\_3 -0.035535 0.247587 0.646538 0.570535 0.010499 -0.032816   
pixel\_0\_4 0.016959 -0.014930 0.053984 0.075355 -0.068580 -0.217234   
... ... ... ... ... ... ...   
pixel\_7\_4 -0.004556 -0.060110 -0.126268 0.110864 1.000000 0.672410   
pixel\_7\_5 -0.000368 -0.027860 -0.075534 -0.065122 0.672410 1.000000   
pixel\_7\_6 -0.016377 0.030730 0.021227 -0.018654 0.383310 0.786449   
pixel\_7\_7 -0.006021 -0.013251 0.072617 0.102549 0.111429 0.311382   
target -0.020572 -0.036070 0.016617 -0.071592 -0.112225 -0.104000   
  
 pixel\_7\_6 pixel\_7\_7 target   
pixel\_0\_0 NaN NaN NaN   
pixel\_0\_1 0.062895 0.001071 -0.048067   
pixel\_0\_2 0.033075 0.109046 0.003901   
pixel\_0\_3 0.045415 0.119529 -0.067370   
pixel\_0\_4 -0.229098 -0.183080 0.115201   
... ... ... ...   
pixel\_7\_4 0.383310 0.111429 -0.112225   
pixel\_7\_5 0.786449 0.311382 -0.104000   
pixel\_7\_6 1.000000 0.483112 -0.059124   
pixel\_7\_7 0.483112 1.000000 -0.168184   
target -0.059124 -0.168184 1.000000   
  
[65 rows x 65 columns]

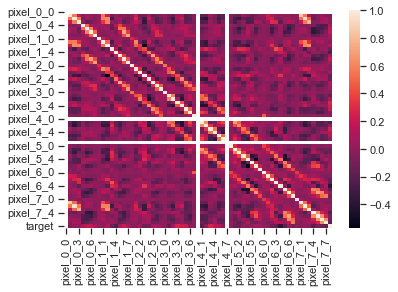
В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

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>

sns.heatmap(data.corr())

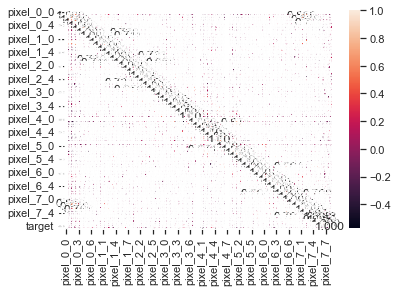
<AxesSubplot:>



data2 = data  
data2.drop(data2.columns[[1,2,3,4,5,11,12,13,14,15,16,17]], axis = 1)  
print(data2)  
# Вывод значений в ячейках  
sns.heatmap(data2.corr(), annot=True, fmt='.3f')

pixel\_0\_0 pixel\_0\_1 pixel\_0\_2 pixel\_0\_3 pixel\_0\_4 pixel\_0\_5 \  
0 0.0 0.0 5.0 13.0 9.0 1.0   
1 0.0 0.0 0.0 12.0 13.0 5.0   
2 0.0 0.0 0.0 4.0 15.0 12.0   
3 0.0 0.0 7.0 15.0 13.0 1.0   
4 0.0 0.0 0.0 1.0 11.0 0.0   
... ... ... ... ... ... ...   
1792 0.0 0.0 4.0 10.0 13.0 6.0   
1793 0.0 0.0 6.0 16.0 13.0 11.0   
1794 0.0 0.0 1.0 11.0 15.0 1.0   
1795 0.0 0.0 2.0 10.0 7.0 0.0   
1796 0.0 0.0 10.0 14.0 8.0 1.0   
  
 pixel\_0\_6 pixel\_0\_7 pixel\_1\_0 pixel\_1\_1 ... pixel\_6\_7 pixel\_7\_0 \  
0 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1 0.0 0.0 0.0 0.0 ... 0.0 0.0   
2 0.0 0.0 0.0 0.0 ... 0.0 0.0   
3 0.0 0.0 0.0 8.0 ... 0.0 0.0   
4 0.0 0.0 0.0 0.0 ... 0.0 0.0   
... ... ... ... ... ... ... ...   
1792 0.0 0.0 0.0 1.0 ... 0.0 0.0   
1793 1.0 0.0 0.0 0.0 ... 0.0 0.0   
1794 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1795 0.0 0.0 0.0 0.0 ... 0.0 0.0   
1796 0.0 0.0 0.0 2.0 ... 0.0 0.0   
  
 pixel\_7\_1 pixel\_7\_2 pixel\_7\_3 pixel\_7\_4 pixel\_7\_5 pixel\_7\_6 \  
0 0.0 6.0 13.0 10.0 0.0 0.0   
1 0.0 0.0 11.0 16.0 10.0 0.0   
2 0.0 0.0 3.0 11.0 16.0 9.0   
3 0.0 7.0 13.0 13.0 9.0 0.0   
4 0.0 0.0 2.0 16.0 4.0 0.0   
... ... ... ... ... ... ...   
1792 0.0 2.0 14.0 15.0 9.0 0.0   
1793 0.0 6.0 16.0 14.0 6.0 0.0   
1794 0.0 2.0 9.0 13.0 6.0 0.0   
1795 0.0 5.0 12.0 16.0 12.0 0.0   
1796 1.0 8.0 12.0 14.0 12.0 1.0   
  
 pixel\_7\_7 target   
0 0.0 0.0   
1 0.0 1.0   
2 0.0 2.0   
3 0.0 3.0   
4 0.0 4.0   
... ... ...   
1792 0.0 9.0   
1793 0.0 0.0   
1794 0.0 8.0   
1795 0.0 9.0   
1796 0.0 8.0   
  
[1797 rows x 65 columns]

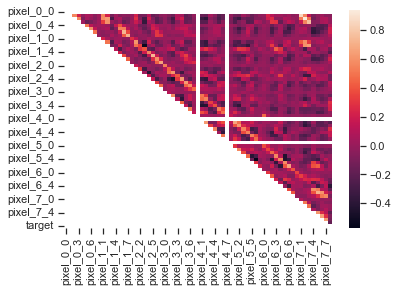
<AxesSubplot:>



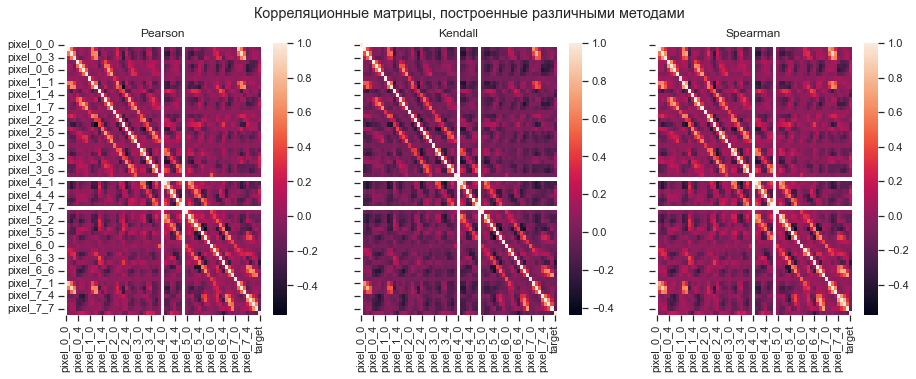
# Треугольный вариант матрицы  
mask = np.zeros\_like(data.corr(), dtype=np.bool)  
# чтобы оставить нижнюю часть матрицы  
# mask[np.triu\_indices\_from(mask)] = True  
# чтобы оставить верхнюю часть матрицы  
mask[np.tril\_indices\_from(mask)] = True  
sns.heatmap(data.corr(), mask=mask, fmt='.3f')

C:\Users\Pocht\AppData\Local\Temp/ipykernel\_20988/316288358.py:2: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool\_` here.  
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations  
 mask = np.zeros\_like(data.corr(), dtype=np.bool)

<AxesSubplot:>



fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))  
sns.heatmap(data.corr(method='pearson'), ax=ax[0], fmt='.2f')  
sns.heatmap(data.corr(method='kendall'), ax=ax[1], fmt='.2f')  
sns.heatmap(data.corr(method='spearman'), ax=ax[2], fmt='.2f')  
fig.suptitle('Корреляционные матрицы, построенные различными методами')  
ax[0].title.set\_text('Pearson')  
ax[1].title.set\_text('Kendall')  
ax[2].title.set\_text('Spearman')



Необходимо отметить, что тепловая карта не очень хорошо подходит для определения корреляции нецелевых признаков между собой.

Но в реальной модели могут быть сотни признаков и коррелирующие признаки могут образовывать группы, состояшие более чем из двух признаков. Увидеть такие группы с помощью тепловой карты сложно.