Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Технологии машинного обучения»

Отчет по лабораторной работе №1

Выполнила:

студент группы ИУ5-63Б

Латыпова К.Н.

Проверил:

преподаватель каф. ИУ5

Гапанюк Ю.Е.

Москва, 2020 г.

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.
3. Визуальное исследование датасета.
4. Информация о корреляции признаков.

- Сформировать отчет и разместить его в своем репозитории на github.

Текст программы и экранные формы:

ЛР №1

1) Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных о ценах на жилье в Бостоне:

Содержкание:

CRIM per capita crime rate by town - Уровень преступности на душу населения по городам

ZN proportion of residential land zoned for lots over 25,000 sq.ft. - ЗН доля земли под жилую застройку зонирована на участки площадью более 25 000 кв. Футов.

INDUS proportion of non-retail business acres per town - INDUS доля акров, не относящихся к розничной торговле, на город

CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) - Фиктивная переменная CHAS Charles River (= 1, если тракт ограничивает реку; 0 в противном случае)

NOX nitric oxides concentration (parts per 10 million) - Концентрация оксидов азота NOX (частей на 10 млн.)

RM average number of rooms per dwelling - RM среднее количество комнат в одном жилом помещении

AGE proportion of owner-occupied units built prior to 1940 - ВОЗРАСТНАЯ доля занятых владельцами квартир, построенных до 1940 года

DIS weighted distances to five Boston employment centres - DIS взвешенные расстояния до пяти бостонских центров занятости

RAD index of accessibility to radial highways - РАД индекс доступности радиальных магистралей

TAX full-value property-tax rate per 10,000 dol. - НАЛОГ на недвижимость с полной стоимостью-ставка налога на 10 000 долларов США

PTRATIO pupil-teacher ratio by town - PTRATIO соотношение учеников и учителей по городам

B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town - B 1000(Bk - 0.63)^2, где Bk-доля чернокожих по городам

LSTAT % lower status of the population - LSTAT % более низкий статус населения

MEDV Median value of owner-occupied homes in 1000's dol. - MEDV Медианная стоимость домов, занятых владельцами, в 1000-х годах

Импорт библиотек

In [52]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
import warnings
from sklearn import datasets
from sklearn.datasets import load_boston
from sklearn import linear_model
from sklearn.cluster import KMeans
from sklearn import metrics
from pandas import DataFrame
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Загрузка данных

In [53]:
```
boston = load_boston()
data = pd.DataFrame(boston.data, columns=boston.feature_names)
data['TARGET'] = boston.target
```

2)Основные характеристики датасета

In [54]:
```
# Первые пять строк датасета
data.head()
```

Out[54]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGET
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

In [55]:
```
# Размер датасета
data.shape
```

Out[55]:
```
(506, 14)
```

In [56]:
```
total_count = data.shape[0]
print('Всего строк: {}'.format(total_count))
```

```
Всего строк: 506
```

In [57]:
```
# Список колонок
data.columns
```

Out[57]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'TARGET'],
      dtype='object')
```

In [58]:
```
# Список колонок с типами данных
data.dtypes
```

Out[58]:
```
CRIM       float64
ZN         float64
INDUS      float64
CHAS       float64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD        float64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
TARGET     float64
dtype: object
```

В [59]:
```
# 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))
```

```
CRIM - 0
ZN - 0
INDUS - 0
CHAS - 0
NOX - 0
RM - 0
AGE - 0
DIS - 0
RAD - 0
TAX - 0
PTRATIO - 0
B - 0
LSTAT - 0
TARGET - 0
```

В [60]:
```
# Основные статистические характеристки набора данных
data.describe()
```

Out[60]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.0
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.6
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.1
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.7
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.9
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.3
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.9
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.9

В [61]:
```
# Определим уникальные значения для целевого признака
data['TARGET'].unique()
```

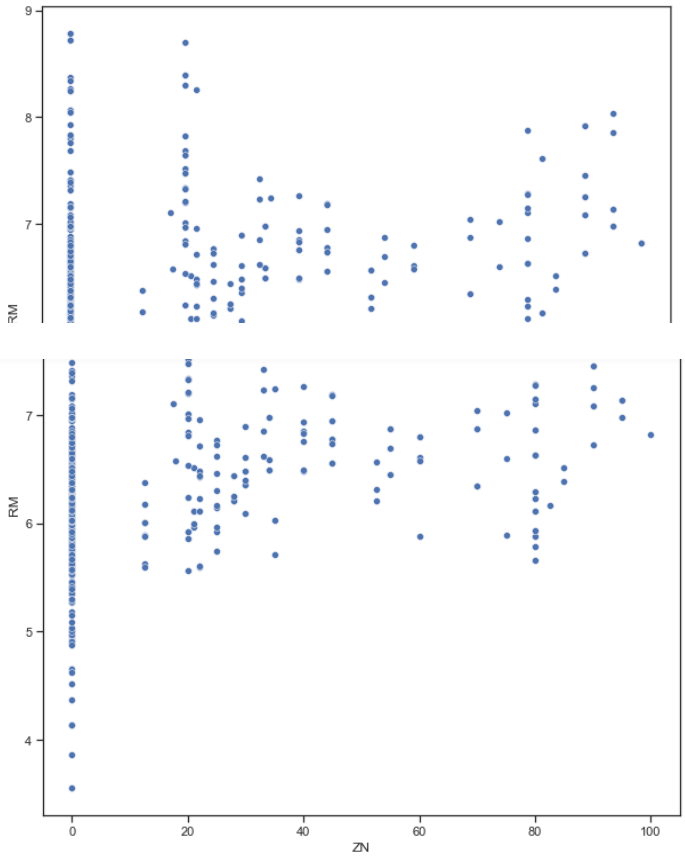
Out[61]:
```
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
       21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 13.6, 19.6, 15.2, 14.5,
       15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 13.2, 13.1, 13.5, 20. ,
       24.7, 30.8, 34.9, 26.6, 25.3, 21.2, 19.3, 14.4, 19.4, 19.7, 20.5,
       25. , 23.4, 35.4, 31.6, 23.3, 18.7, 16. , 22.2, 33. , 23.5, 22. ,
       17.4, 20.9, 24.2, 22.8, 24.1, 21.4, 20.8, 20.3, 28. , 23.9, 24.8,
       22.5, 23.6, 22.6, 20.6, 28.4, 38.7, 43.8, 33.2, 27.5, 26.5, 18.6,
       20.1, 19.5, 19.8, 18.8, 18.5, 18.3, 19.2, 17.3, 15.7, 16.2, 18. ,
       14.3, 23. , 18.1, 17.1, 13.3, 17.8, 14. , 13.4, 11.8, 13.8, 14.6,
       15.4, 21.5, 15.3, 17. , 41.3, 24.3, 27. , 50. , 22.7, 23.8, 22.3,
       19.1, 29.4, 23.2, 24.6, 29.9, 37.2, 39.8, 37.9, 32.5, 26.4, 29.6,
       32. , 29.8, 37. , 30.5, 36.4, 31.1, 29.1, 33.3, 30.3, 34.6, 32.9,
       42.3, 48.5, 24.4, 22.4, 28.1, 23.7, 26.7, 30.1, 44.8, 37.6, 46.7,
       31.5, 31.7, 41.7, 48.3, 29. , 25.1, 17.6, 24.5, 26.2, 42.8, 21.9,
       44. , 36. , 33.8, 43.1, 48.8, 31. , 36.5, 30.7, 43.5, 20.7, 21.1,
       25.2, 35.2, 32.4, 33.1, 35.1, 45.4, 46. , 32.2, 28.5, 37.3, 27.9,
       28.6, 36.1, 28.2, 16.1, 22.1, 19. , 32.7, 31.2, 17.2, 16.8, 10.2,
       10.4, 10.9, 11.3, 12.3,  8.8,  7.2, 10.5,  7.4, 11.5, 15.1,  9.7,
       12.5,  8.5,  5. ,  6.3,  5.6, 12.1,  8.3, 11.9, 17.9, 16.3,  7. ,
        7.5,  8.4, 16.7, 14.2, 11.7, 11. ,  9.5, 14.1,  9.6,  8.7, 12.8,
       10.8, 14.9, 12.6, 13. , 16.4, 17.7, 12. , 21.8,  8.1])
```

3) Визуальное исследование датасета

Диаграмма рассеяния

B [62]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='ZN', y='RM', data=data)
```

Out[62]:
```
<AxesSubplot:xlabel='ZN', ylabel='RM'>
```

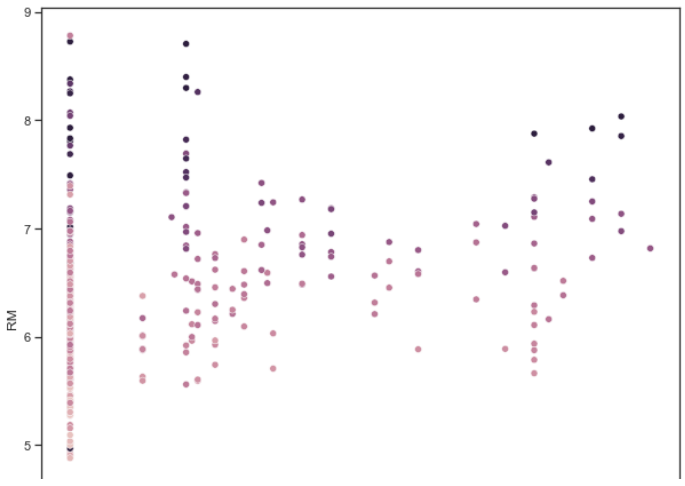


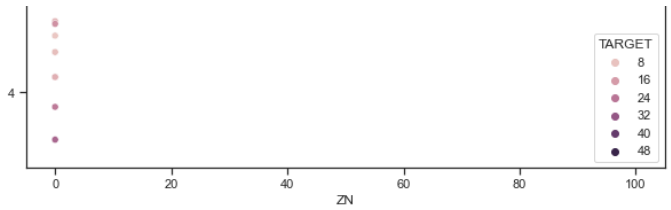
Можно видеть что между полями ZN и RM не пристутствует линейной зависимости.

Посмотрим насколько на эту зависимость влияет целевой признак.

B [63]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='ZN', y='RM', data=data, hue='TARGET')
```

Out[63]:
```
<AxesSubplot:xlabel='ZN', ylabel='RM'>
```



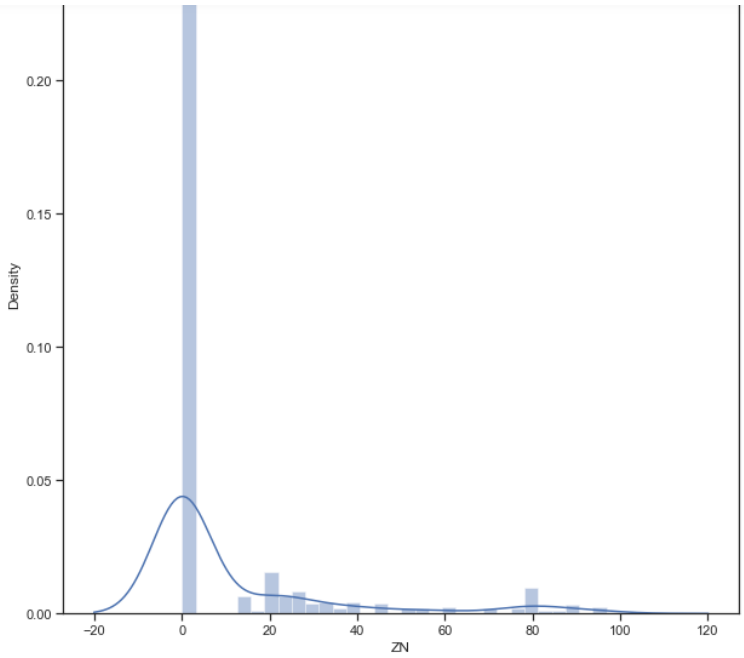


Гистограмма

B [64]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.distplot(data['ZN'])
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distributions.py:2551: 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)
```

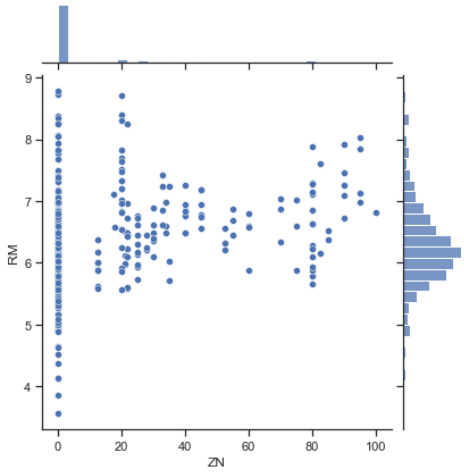
Out[64]: <AxesSubplot:xlabel='ZN', ylabel='Density'>



Jointplot

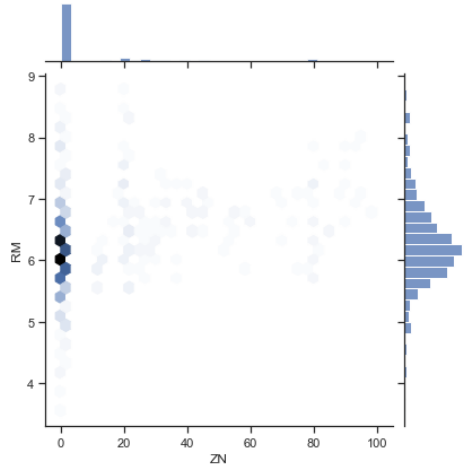
B [65]:
```
sns.jointplot(x='ZN', y='RM', data=data)
```

Out[65]: <seaborn.axisgrid.JointGrid at 0x2bed5aead60>



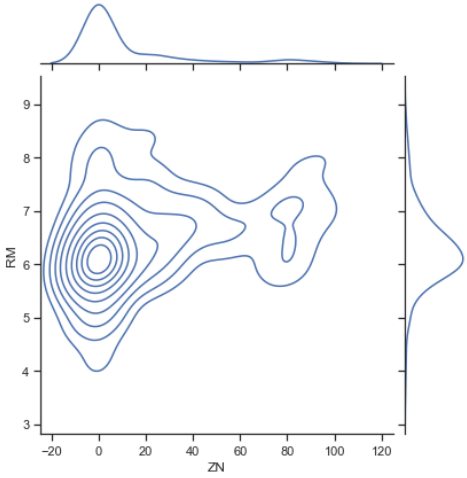
B [66]:
```
sns.jointplot(x='ZN', y='RM', data=data, kind="hex")
```

Out[66]:
```
<seaborn.axisgrid.JointGrid at 0x2bed6034190>
```



B [67]:
```
sns.jointplot(x='ZN', y='RM', data=data, kind="kde")
```

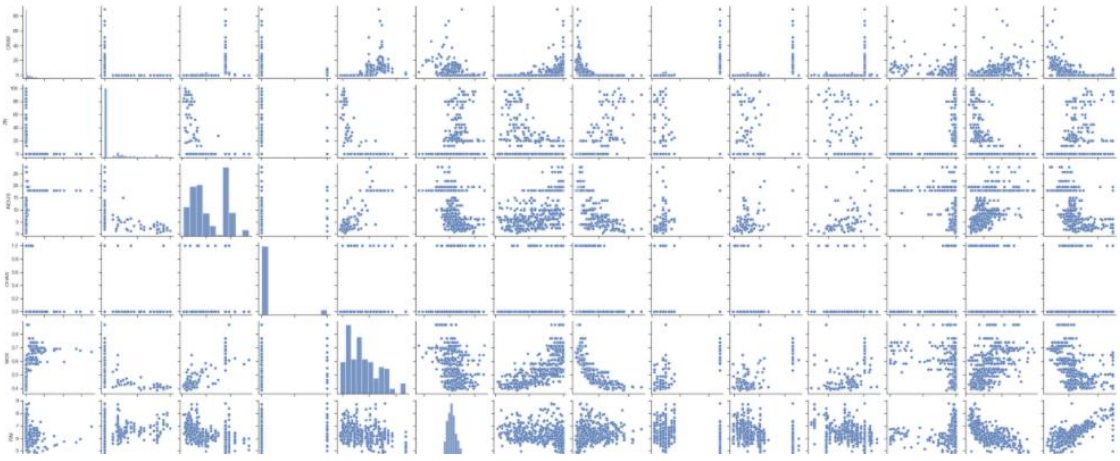
Out[67]:
```
<seaborn.axisgrid.JointGrid at 0x2bed6201a60>
```

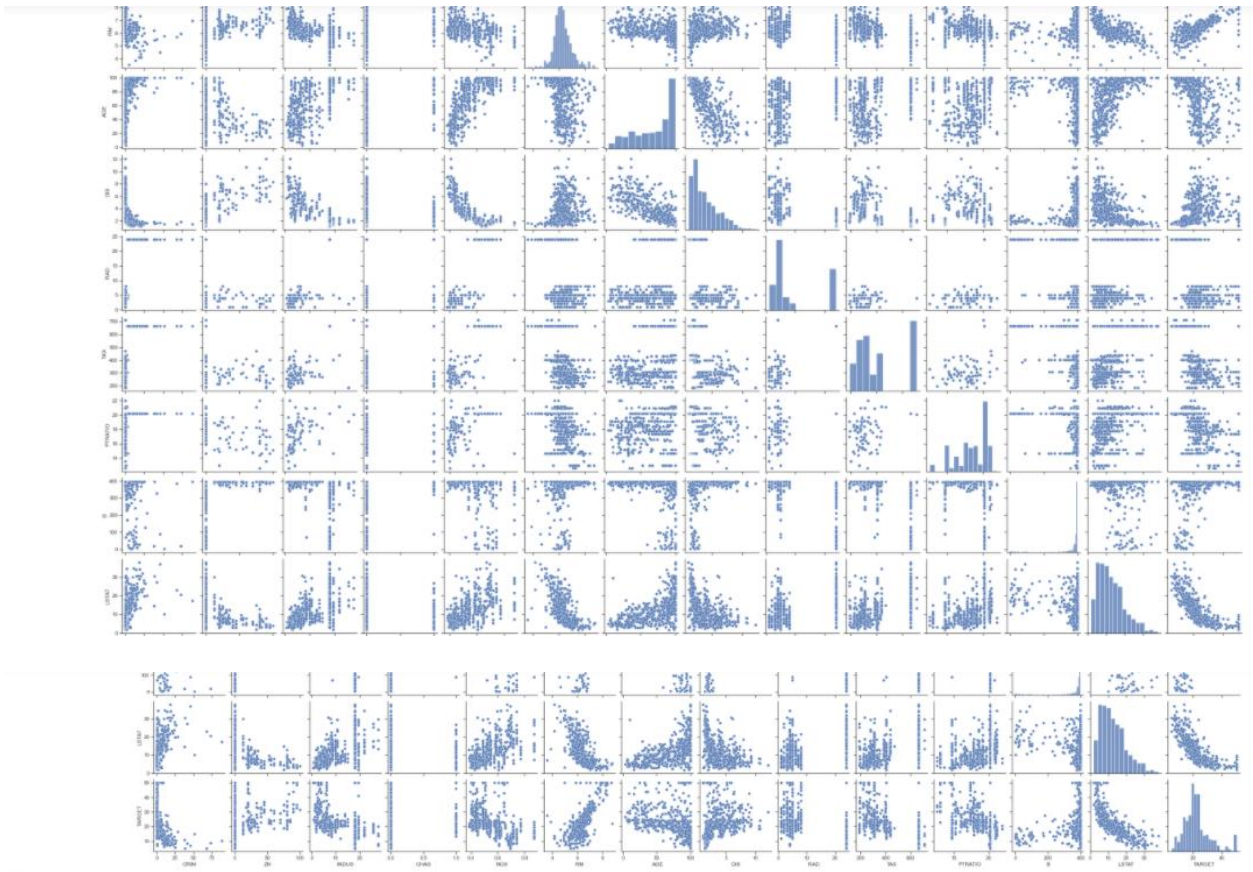


"Парные диаграммы"

B [68]:
```
sns.pairplot(data)
```

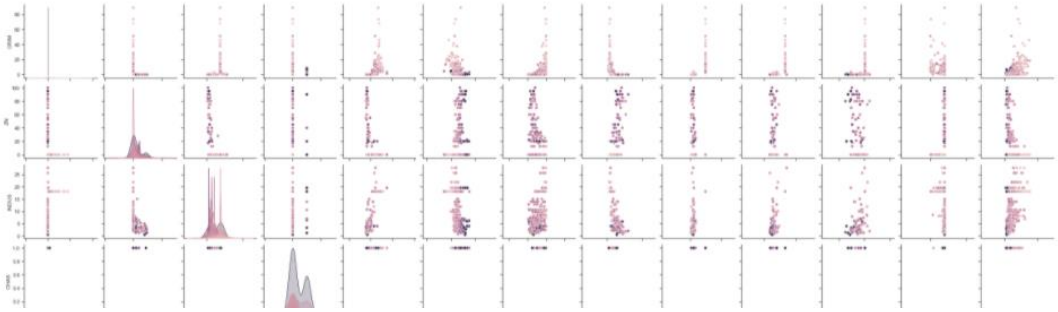
Out[68]:
```
<seaborn.axisgrid.PairGrid at 0x2bed62726a0>
```



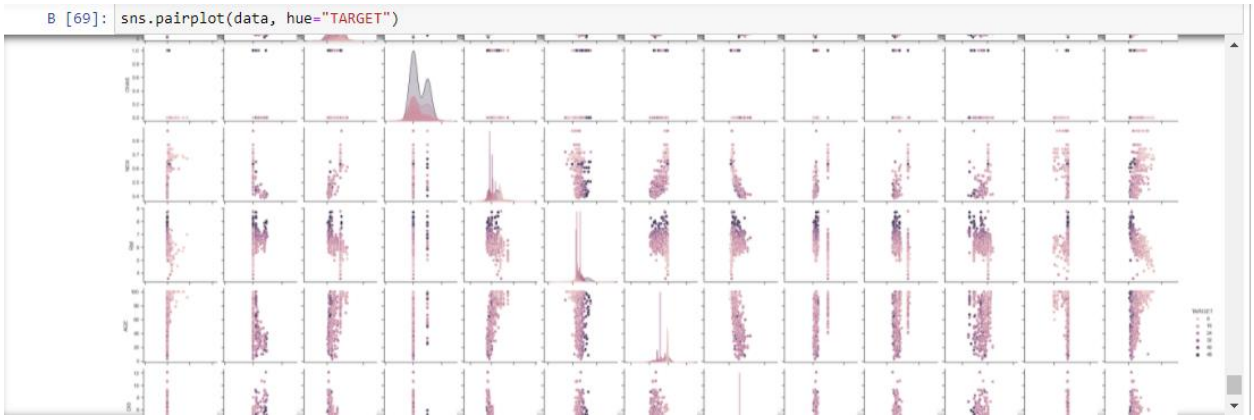


```
B [69]: sns.pairplot(data, hue="TARGET")
```

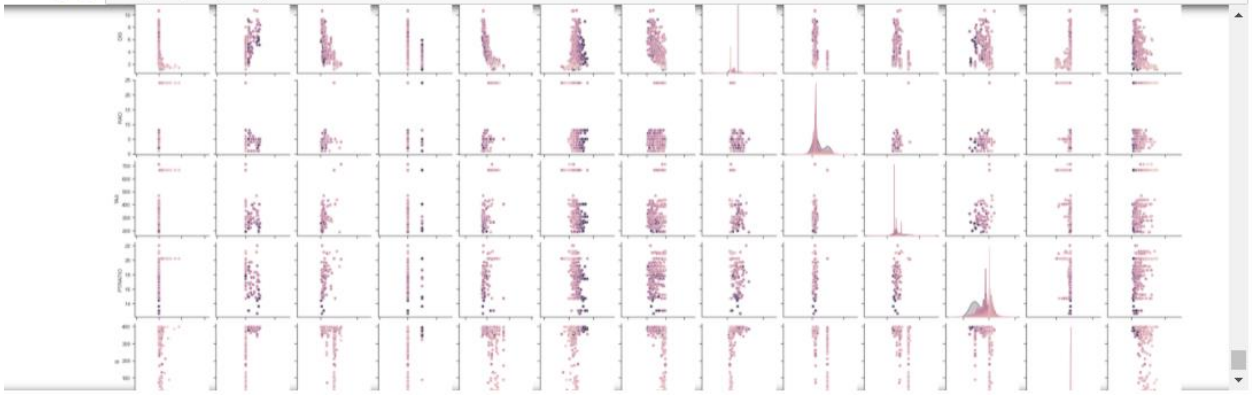
```
Out[69]: <seaborn.axisgrid.PairGrid at 0x2bee0d3ae20>
```



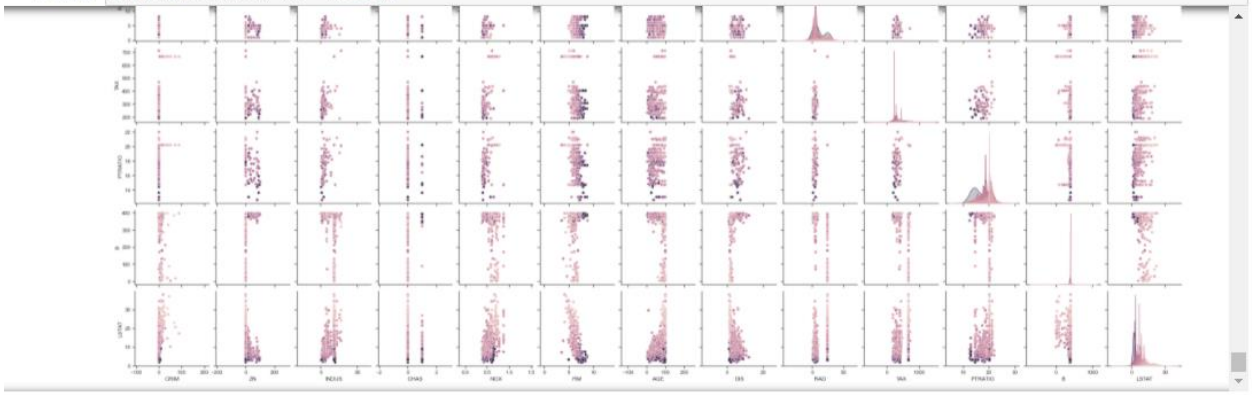
```
B [69]: sns.pairplot(data, hue="TARGET")
```



```
В [69]: sns.pairplot(data, hue="TARGET")
```



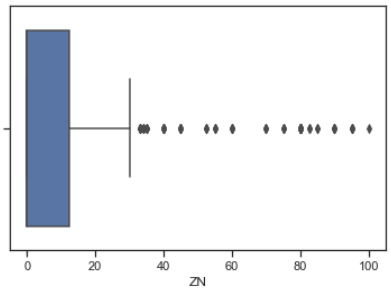

```
В [69]: sns.pairplot(data, hue="TARGET")
```

Ящик с усами

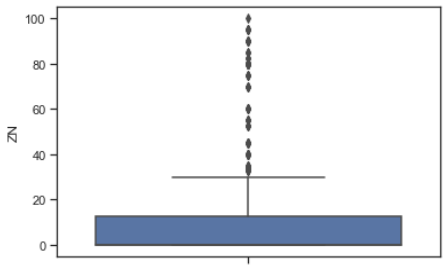
```
В [70]: sns.boxplot(x=data['ZN'])
```

```
Out[70]: <AxesSubplot:xlabel='ZN'>
```



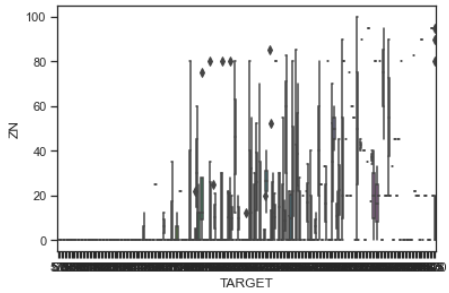

```
В [71]: # По вертикали
        sns.boxplot(y=data['ZN'])
```

```
Out[71]: <AxesSubplot:ylabel='ZN'>
```



В [72]:
```
# Распределение параметра ZN сгруппированные по TARGET.
sns.boxplot(x='TARGET', y='ZN', data=data)
```

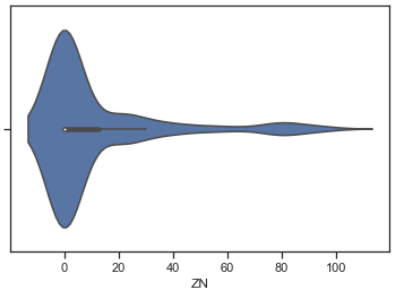
Out[72]: <AxesSubplot:xlabel='TARGET', ylabel='ZN'>



Violin plot

В [73]:
```
sns.violinplot(x=data['ZN'])
```

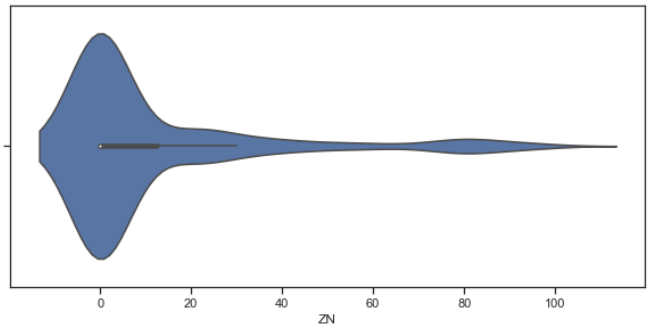
Out[73]: <AxesSubplot:xlabel='ZN'>

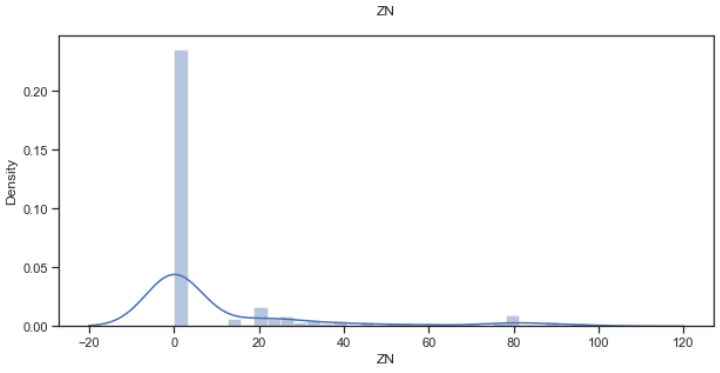


В [74]:
```
fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=data['ZN'])
sns.distplot(data['ZN'], ax=ax[1])
```

```
C:\Users\user\anaconda3\lib\site-packages\seaborn\distributions.py:2551: 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 flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[74]: <AxesSubplot:xlabel='ZN', ylabel='Density'>

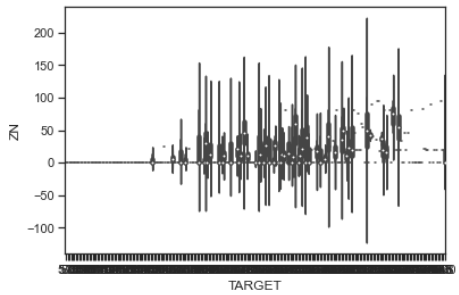




Из приведенных графиков видно, что violinplot действительно показывает распределение плотности.

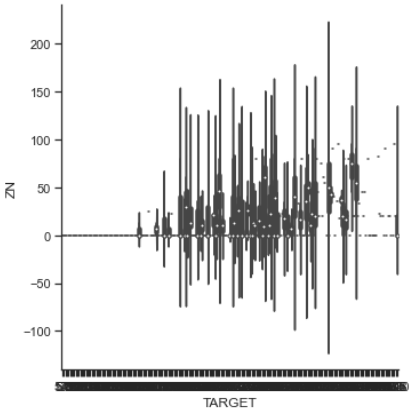
В [75]:
```
# Распределение параметра ZN сгруппированные по TARGET.
sns.violinplot(x='TARGET', y='ZN', data=data)
```

Out[75]: <AxesSubplot:xlabel='TARGET', ylabel='ZN'>



В [76]:
```
sns.catplot(y='ZN', x='TARGET', data=data, kind="violin", split=True)
```

Out[76]: <seaborn.axisgrid.FacetGrid at 0x2beedbaa4f0>



4) Информация о корреляции признаков

В [77]:
```
data.corr()
```

Out[77]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGE
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.38830
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.36044
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.48372
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.17526
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.42732
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.69536
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.37695
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.24992
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.38162
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.46853
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.50778
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.33346
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.73766
TARGET	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.00000

B [78]:
```
data.corr(method='pearson')
```

Out[78]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGE
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.38830
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.36044
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.48372
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.17526
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.42732
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.69536
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.37695
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.24992
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.38162
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.46853
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.50778
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.33346
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.73766
TARGET	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.00000

B [79]:
```
data.corr(method='kendall')
```

Out[79]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGE
CRIM	1.000000	-0.462057	0.521014	0.033948	0.603361	-0.211718	0.497297	-0.539878	0.563969	0.544956	0.312768	-0.264378	0.454837	-0.40396
ZN	-0.462057	1.000000	-0.535468	-0.039419	-0.511464	0.278134	-0.429389	0.478524	-0.234663	-0.289911	-0.361607	0.128177	-0.386818	0.33998
INDUS	0.521014	-0.535468	1.000000	0.075889	0.612030	-0.291318	0.489070	-0.565137	0.353967	0.483228	0.336612	-0.192017	0.465980	-0.41843
CHAS	0.033948	-0.039419	0.075889	1.000000	0.056387	0.048080	0.055616	-0.065619	0.021739	-0.037655	-0.115694	-0.033277	-0.041344	0.11520
NOX	0.603361	-0.511464	0.612030	0.056387	1.000000	-0.215633	0.589608	-0.683930	0.434828	0.453258	0.278678	-0.202430	0.452005	-0.39499
RM	-0.211718	0.278134	-0.291318	0.048080	-0.215633	1.000000	-0.187611	0.179801	-0.076569	-0.190532	-0.223194	0.032951	-0.468231	0.48282
AGE	0.497297	-0.429389	0.489070	0.055616	0.589608	-0.187611	1.000000	-0.609836	0.306201	0.360311	0.251857	-0.154056	0.485359	-0.38775
DIS	-0.539878	0.478524	-0.565137	-0.065619	-0.683930	0.179801	-0.609836	1.000000	-0.361892	-0.381988	-0.223486	0.168631	-0.409347	0.31311
RAD	0.563969	-0.234663	0.353967	0.021739	0.434828	-0.076569	0.306201	-0.361892	1.000000	0.558107	0.251913	-0.214364	0.287943	-0.24811
TAX	0.544956	-0.289911	0.483228	-0.037655	0.453258	-0.190532	0.360311	-0.381988	0.558107	1.000000	0.287769	-0.241606	0.384191	-0.41465
PTRATIO	0.312768	-0.361607	0.336612	-0.115694	0.278678	-0.223194	0.251857	-0.223486	0.251913	0.287769	1.000000	-0.042152	0.330335	-0.39878
B	-0.264378	0.128177	-0.192017	-0.033277	-0.202430	0.032951	-0.154056	0.168631	-0.214364	-0.241606	-0.042152	1.000000	-0.145430	0.12695
LSTAT	0.454837	-0.386818	0.465980	-0.041344	0.452005	-0.468231	0.485359	-0.409347	0.287943	0.384191	0.330335	-0.145430	1.000000	-0.66865
TARGET	-0.403964	0.339989	-0.418430	0.115202	-0.394995	0.482829	-0.387758	0.313115	-0.248115	-0.414650	-0.398789	0.126955	-0.668656	1.00000

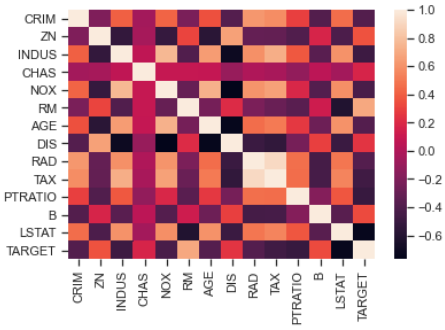
B [80]:
```
data.corr(method='spearman')
```

Out[80]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	TARGE
CRIM	1.000000	-0.571660	0.735524	0.041537	0.821465	-0.309116	0.704140	-0.744986	0.727807	0.729045	0.465283	-0.360555	0.634760	-0.55889
ZN	-0.571660	1.000000	-0.642811	-0.041937	-0.634828	0.361074	-0.544423	0.614627	-0.278767	-0.371394	-0.448475	0.163135	-0.490074	0.43817
INDUS	0.735524	-0.642811	1.000000	0.089841	0.791189	-0.415301	0.679487	-0.757080	0.455507	0.664361	0.433710	-0.285840	0.638747	-0.57825
CHAS	0.041537	-0.041937	0.089841	1.000000	0.068426	0.058813	0.067792	-0.080248	0.024579	-0.044486	-0.136065	-0.039810	-0.050575	0.14061
NOX	0.821465	-0.634828	0.791189	0.068426	1.000000	-0.310344	0.795153	-0.880015	0.586429	0.649527	0.391309	-0.296662	0.636828	-0.56260
RM	-0.309116	0.361074	-0.415301	0.058813	-0.310344	1.000000	-0.278082	0.263168	-0.107492	-0.271898	-0.312923	0.053660	-0.640832	0.63357
AGE	0.704140	-0.544423	0.679487	0.067792	0.795153	-0.278082	1.000000	-0.801610	0.417983	0.526366	0.355384	-0.228022	0.657071	-0.54756
DIS	-0.744986	0.614627	-0.757080	-0.080248	-0.880015	0.263168	-0.801610	1.000000	-0.495806	-0.574336	-0.322041	0.249595	-0.564262	0.44585
RAD	0.727807	-0.278767	0.455507	0.024579	0.586429	-0.107492	0.417983	-0.495806	1.000000	0.704876	0.318330	-0.282533	0.394322	-0.34677
TAX	0.729045	-0.371394	0.664361	-0.044486	0.649527	-0.271898	0.526366	-0.574336	0.704876	1.000000	0.453345	-0.329843	0.534423	-0.56241
PTRATIO	0.465283	-0.448475	0.433710	-0.136065	0.391309	-0.312923	0.355384	-0.322041	0.318330	0.453345	1.000000	-0.072027	0.467259	-0.55590
B	-0.360555	0.163135	-0.285840	-0.039810	-0.296662	0.053660	-0.228022	0.249595	-0.282533	-0.329843	-0.072027	1.000000	-0.210562	0.18566
LSTAT	0.634760	-0.490074	0.638747	-0.050575	0.636828	-0.640832	0.657071	-0.564262	0.394322	0.534423	0.467259	-0.210562	1.000000	-0.85291
TARGET	-0.558891	0.438179	-0.578255	0.140612	-0.562609	0.633576	-0.547562	0.445857	-0.346776	-0.562411	-0.555905	0.185664	-0.852914	1.00000

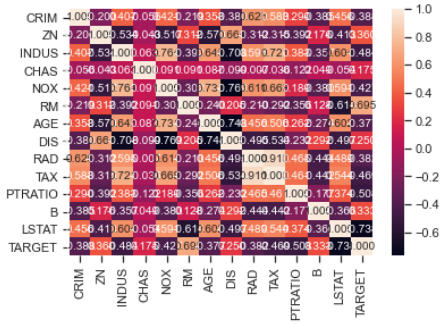
In [81]:
```
sns.heatmap(data.corr())
```

Out[81]: <AxesSubplot:>



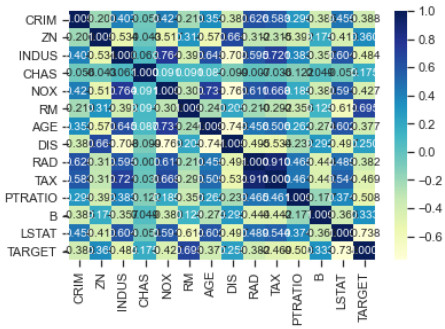
In [82]:
```
# Вывод значений в ячейках
sns.heatmap(data.corr(), annot=True, fmt='.3f')
```

Out[82]: <AxesSubplot:>



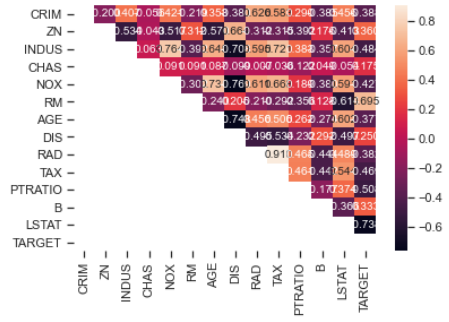
In [83]:
```
# Изменение цветовой гаммы
sns.heatmap(data.corr(), cmap='YlGnBu', annot=True, fmt='.3f')
```

Out[83]: <AxesSubplot:>

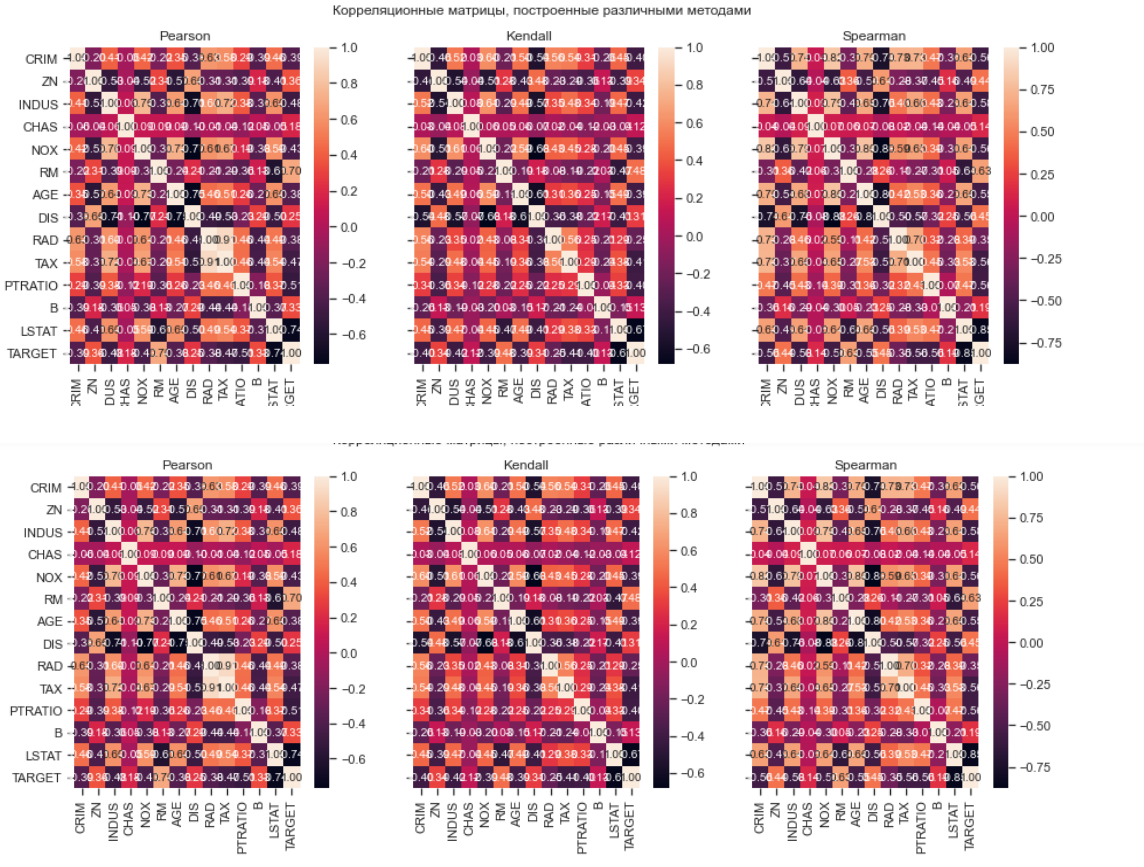


In [84]:
```
# Треугольный вариант матрицы
mask = np.zeros_like(data.corr(), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
mask[np.tril_indices_from(mask)] = True
sns.heatmap(data.corr(), mask=mask, annot=True, fmt='.3f')
```

Out[84]: <AxesSubplot:>



In [85]:
```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(15,5))
sns.heatmap(data.corr(method='pearson'), ax=ax[0], annot=True, fmt='.2f')
sns.heatmap(data.corr(method='kendall'), ax=ax[1], annot=True, fmt='.2f')
sns.heatmap(data.corr(method='spearman'), ax=ax[2], annot=True, 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')
```



In [ ]: