

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

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Теория машинного обучения»

**Отчет по лабораторной работе №1
«Разведочный анализ данных. Исследование и визуализация данных»**

Выполнил:
студент группы ИУ5-63Б
Ветошкин Артём

Подпись и дата:
14.06.22

Проверил:
Юрий Евгеньевич Гапанюк

Подпись и дата:

Москва, 2022 г.

Разведочный анализ данных. Исследование и визуализация данных.

1) Текстовое описание набора данных

В качестве набора данных мы будем использовать набор данных о ценах на жилье в Бостоне - <https://archive.ics.uci.edu/ml/machine-learning-databases/housing/>

Датасет состоит из файла:

- housing.data - обучающая выборка (в этом примере используется только данный файл)

Файл содержит следующие колонки:

- CRIM - уровень преступности на душу населения в разбивке по городам.
- ZN - доля жилой земли, зонированной на участки площадью более 25 000 кв. футов.
- INDUS - доля акров, не связанных с розничной торговлей, на город.
- CHAS - Фиктивная переменная реки Чарльз (= 1, если тракт ограничивает реку; 0 в противном случае).
- NOX - концентрация оксидов азота (частей на 10 миллионов).
- RM - среднее количество комнат в жилом помещении.
- AGE - доля жилых помещений, занятых владельцами, построенных до 1940 года.
- DIS - взвешенные расстояния до пяти бостонских центров занятости.
- RAD - индекс доступности к радиальным магистралям.
- TAX - ставка налога на недвижимость с полной стоимостью за 10 000 долларов США.
- PTRATIO - соотношение учащихся и учителей в разбивке по городам.
- B - $1000(B_k - 0,63)^2$, где B_k - доля чернокожих по городам.
- LSTAT - % более низкий статус населения.
- MEDV - средняя стоимость домов, занятых владельцами, в 1000 долларов.

Импорт библиотек

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

Загрузка данных

Загрузим файлы датасета из библиотеке `sklearn.datasets`, и преобразуем в `DataFrame` библиотеки `Pandas`.

```
ds = load_boston()
df = pd.DataFrame(data= np.c_[ds['data'], ds['target']],
                  columns= list(ds['feature_names']) + ['target'])
```

2) Основные характеристики датасета

```
# Первые 5 строк датасета
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	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

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
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

```
# Размер датасета - 8143 строк, 7 колонок
df.shape
```

```
(506, 14)
```

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

```
Всего строк: 506
```

```
# Список колонок
df.columns
```

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

```
# Список колонок с типами данных
df.dtypes
```

```
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
```

```
# Проверим наличие пустых значений
# Цикл по колонкам датасета
for col in df.columns:
    # Количество пустых значений - все значения заполнены
    temp_null_count = df[df[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
```

```
# Основные статистические характеристики набора данных
df.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

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

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

```
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])
```

Целевой признак имеет множество различных значений

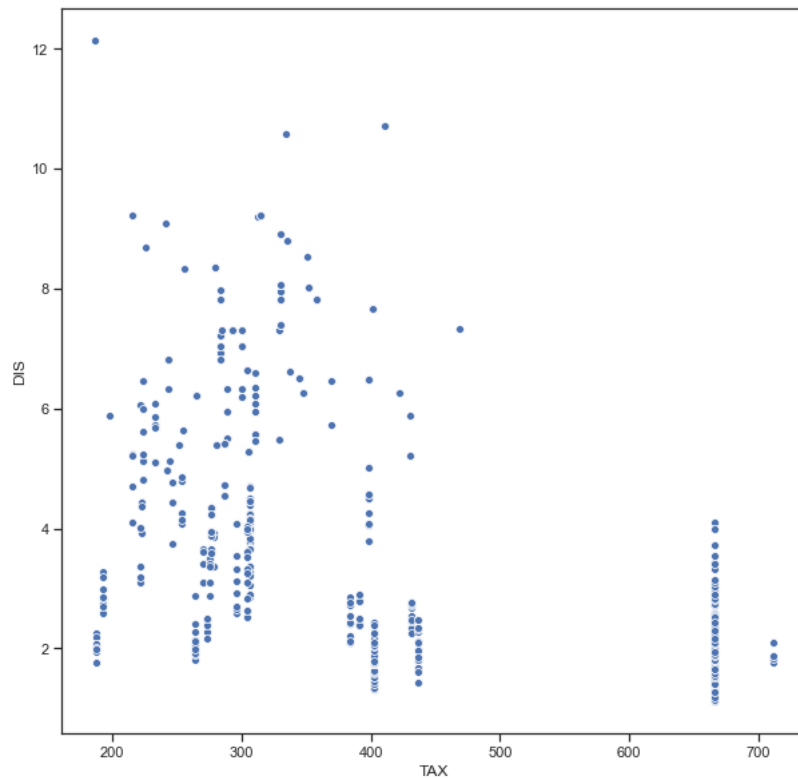
3) Визуальное исследование датасета

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

Диаграмма рассеяния

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='TAX', y='DIS', data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x273fc52c850>
```

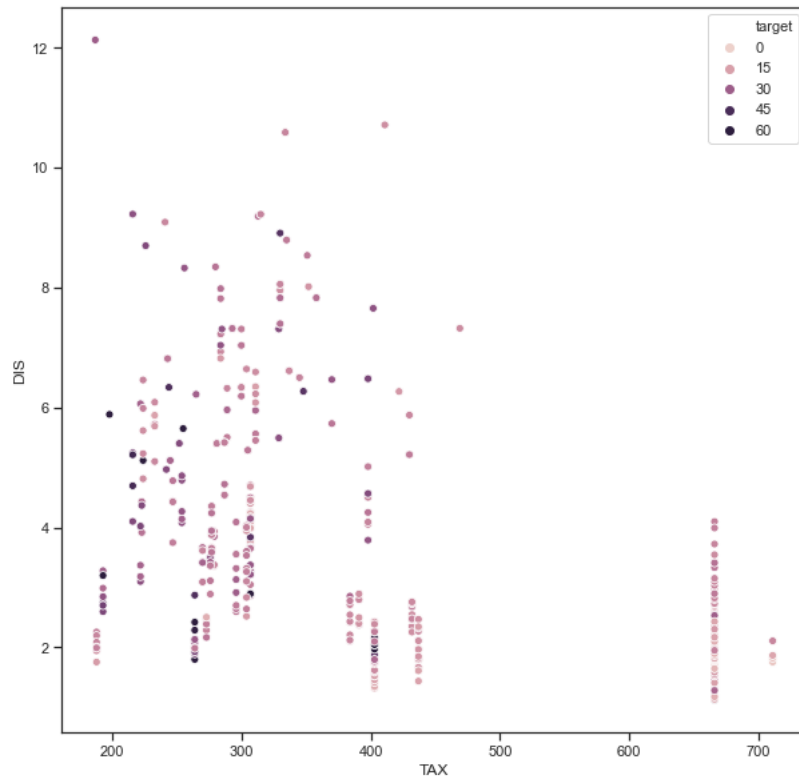


Можно сделать вывод, что налог на недвижимость не имеет ярко выраженной зависимости с удалённостью центра занятости

Посмотрим насколько на эту зависимость влияет целевой признак.

```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='TAX', y='DIS', data=df, hue='target')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x273fcd224f0>
```

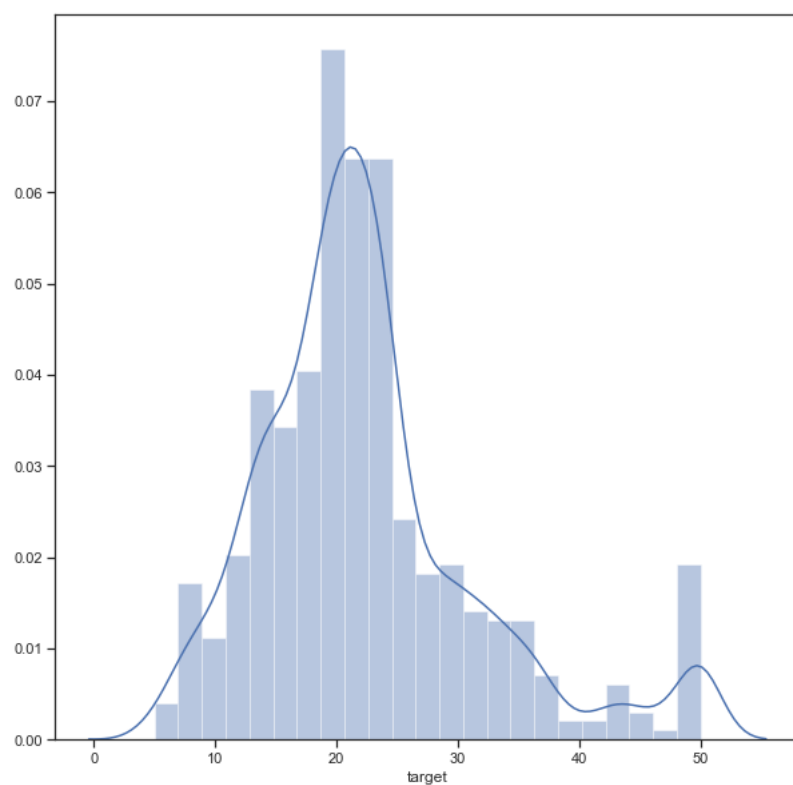


Гистограмма

Позволяет оценить плотность вероятности распределения данных.

```
fig, ax = plt.subplots(figsize=(10,10))  
sns.distplot(df['target'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x273fccdfc70>

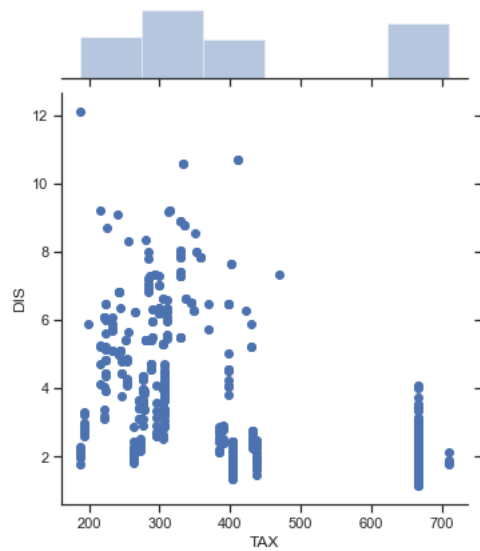


Jointplot

Комбинация гистограмм и диаграмм рассеивания.

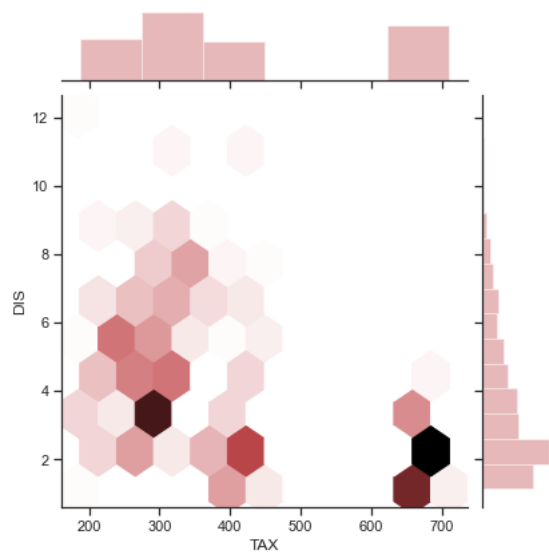
```
sns.jointplot(x='TAX', y='DIS', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x273f7329910>
```



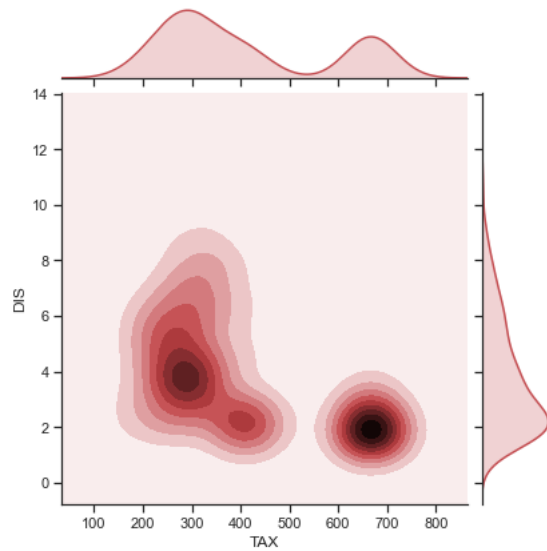
```
sns.jointplot(x='TAX', y='DIS', data=df, kind="hex", color="r")
```

```
<seaborn.axisgrid.JointGrid at 0x273fd15b6a0>
```



```
sns.jointplot(x='TAX', y='DIS', data=df, kind="kde", color="r")
```

```
<seaborn.axisgrid.JointGrid at 0x273fd671640>
```



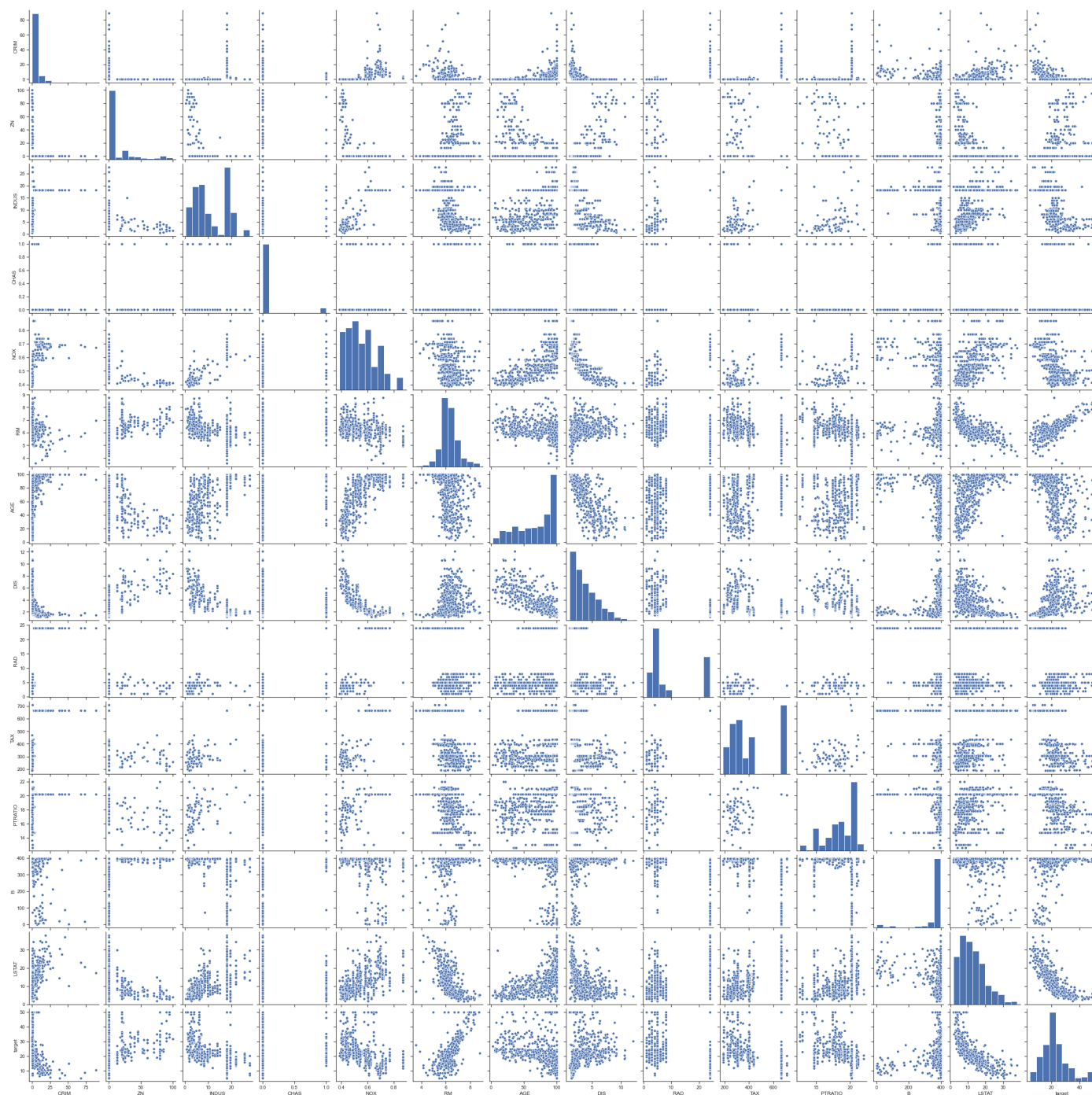
"Парные диаграммы"

Комбинация гистограмм и диаграмм рассеивания для всего набора данных.

Выводится матрица графиков. На пересечении строки и столбца, которые соответствуют двум показателям, строится диаграмма рассеивания. В главной диагонали матрицы строятся гистограммы распределения соответствующих показателей.

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x273fd730e20>
```

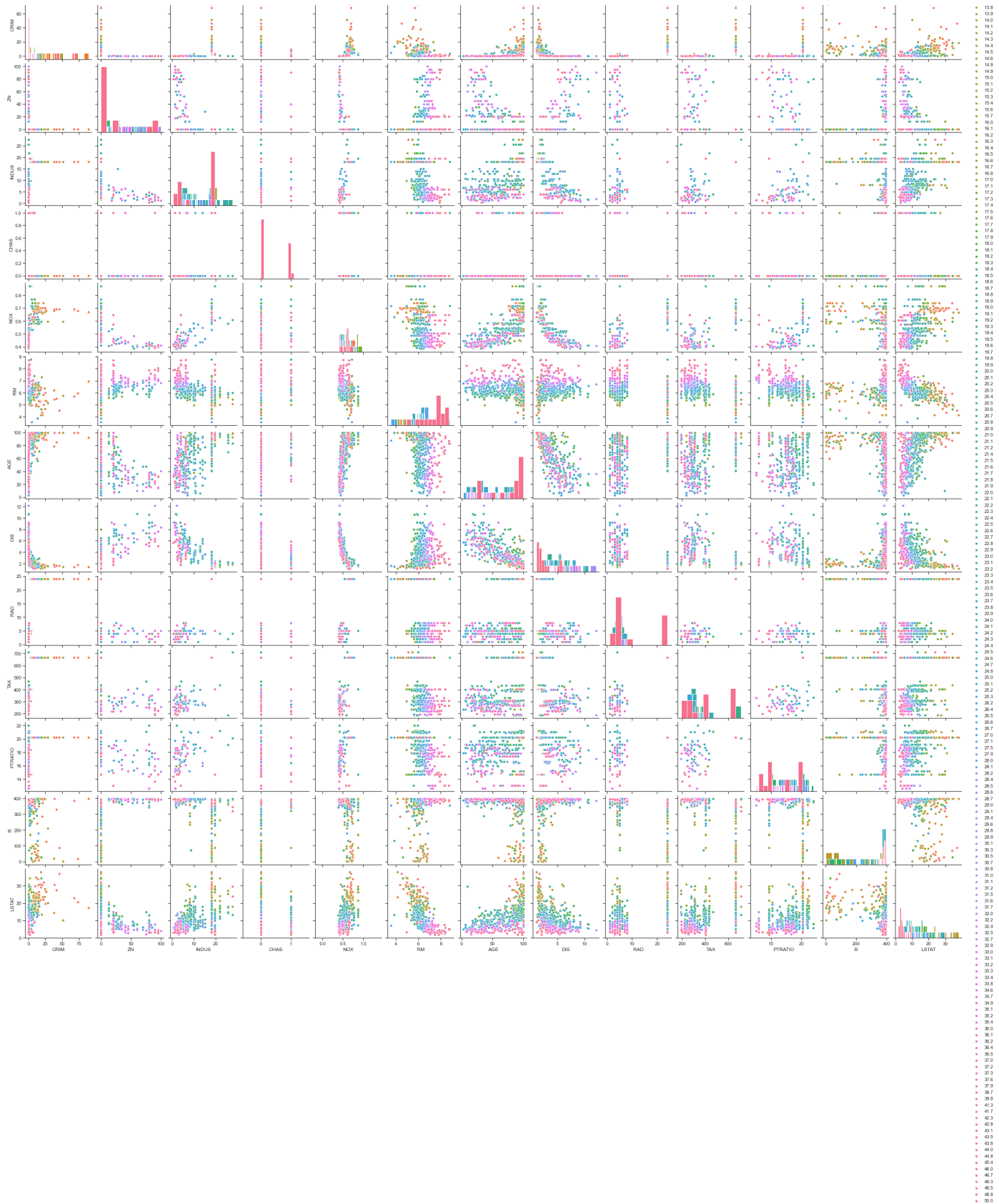
С помощью параметра "hue" возможна группировка по значениям какого-либо признака.

```
sns.pairplot(df, hue="target", diag_kind='hist')
```

```
<seaborn.axisgrid.PairGrid at 0x273c6075700>
```

target

- 5.0
- 5.5
- 6.0
- 6.5
- 7.0
- 7.2
- 7.4
- 7.5
- 8.1
- 8.3
- 8.4
- 8.5
- 8.7
- 8.8
- 9.0
- 9.1
- 9.2
- 9.4
- 9.5
- 9.8
- 10.0
- 10.1
- 10.2
- 10.3
- 10.4
- 10.5
- 10.6
- 10.7
- 10.8
- 10.9
- 11.0
- 11.1
- 11.2
- 11.3
- 11.4
- 11.5
- 11.6
- 11.7
- 11.8
- 11.9
- 12.0
- 12.1
- 12.2
- 12.3
- 12.4
- 12.5
- 12.6
- 12.7
- 12.8
- 12.9
- 13.0
- 13.1
- 13.2
- 13.3
- 13.4
- 13.5
- 13.6

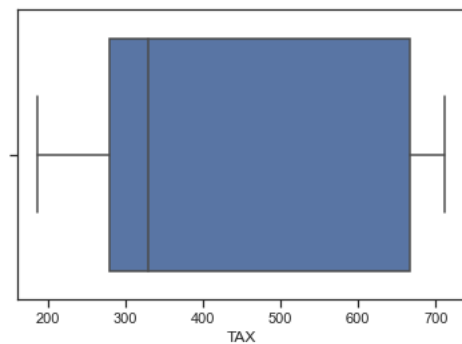


Ящик с усами

Отображает одномерное распределение вероятности.

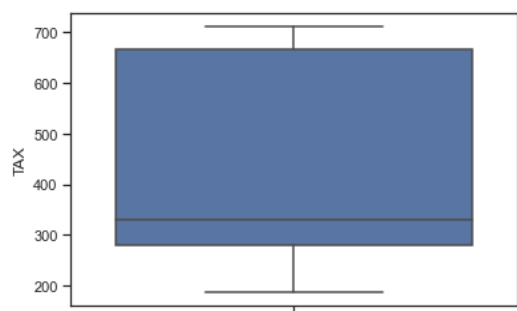
```
sns.boxplot(x=df['TAX'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x27393fc6040>
```

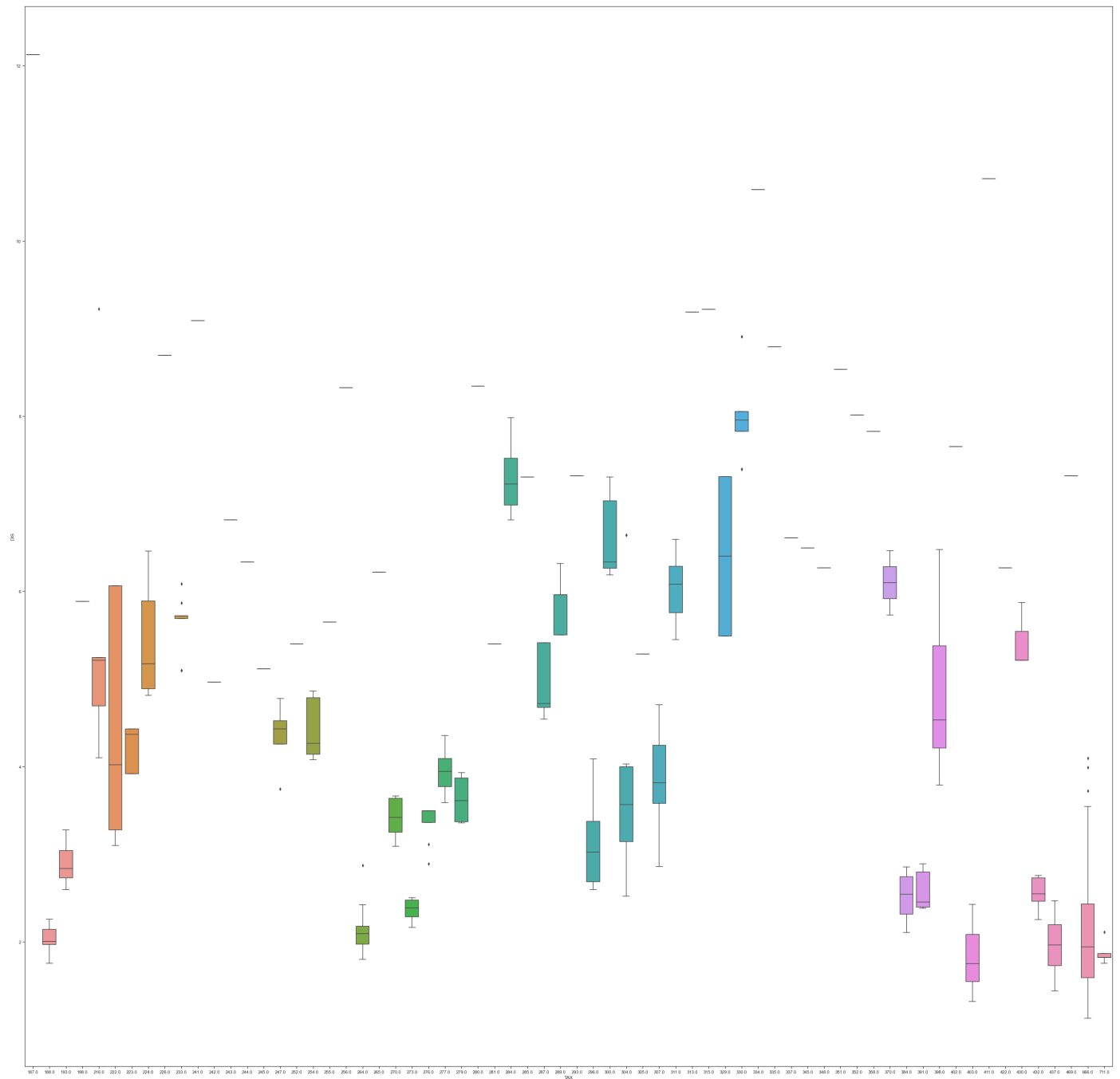


```
# По вертикали  
sns.boxplot(y=df['TAX'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2739307a5b0>
```



```
# Распределение параметра TAX сгруппированные по DIS.  
fig, ax = plt.subplots(figsize=(50,50))  
sns.boxplot(x='TAX', y='DIS', data=df, ax=ax)  
plt.show()
```

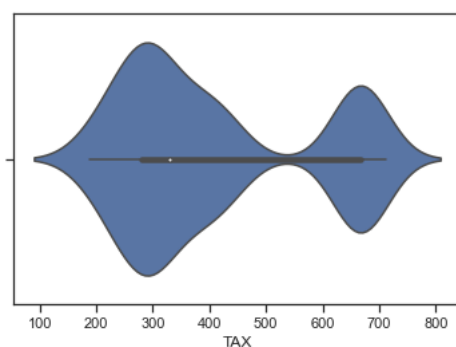


Violin plot

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

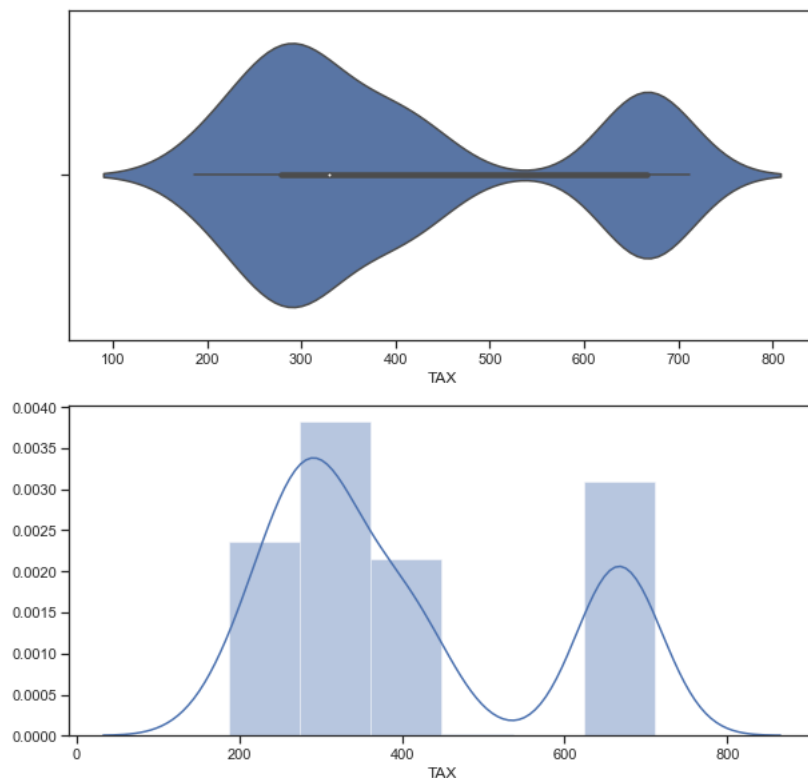
```
sns.violinplot(x=df['TAX'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x273938f4eb0>
```



```
fig, ax = plt.subplots(2, 1, figsize=(10,10))
sns.violinplot(ax=ax[0], x=df['TAX'])
sns.distplot(df['TAX'], ax=ax[1])
```

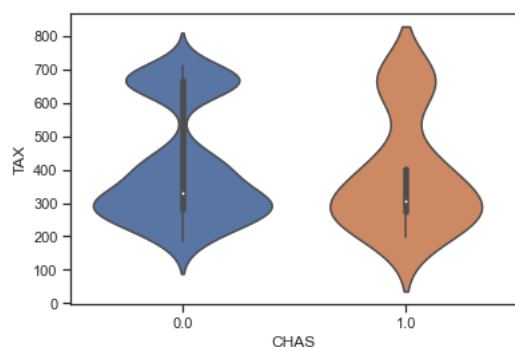
<matplotlib.axes._subplots.AxesSubplot at 0x27395bbf550>



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

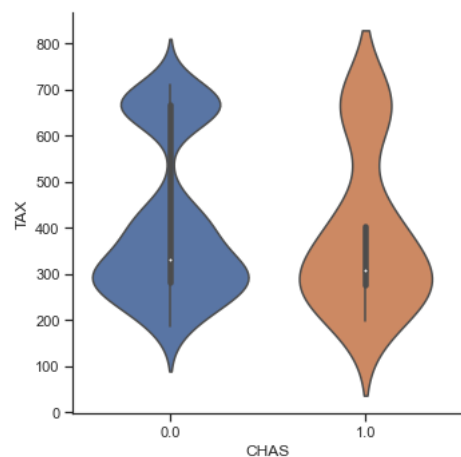
```
# Распределение параметра TAX сгруппированные по CHAS.
sns.violinplot(x='CHAS', y='TAX', data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x273946f5190>



```
sns.catplot(y='TAX', x='CHAS', data=df, kind="violin", split=True)
```

<seaborn.axisgrid.FacetGrid at 0x273945f76a0>



4) Информация о корреляции признаков

Проверка корреляции признаков позволяет решить две задачи:

1. Понять какие признаки (колонки датасета) наиболее сильно коррелируют с целевым признаком (в нашем примере это колонка "RM"). 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.
2. 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.

```
df.corr()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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
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
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
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
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
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
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
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
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366081
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

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 наиболее сильно коррелирует с освещенностью (0.9) и концентрацией углекислого газа (0.71). Эти признаки обязательно следует оставить в модели.
- Целевой признак отчасти коррелирует с температурой (0.54). Этот признак стоит также оставить в модели.
- Целевой признак слабо коррелирует с влажностью (0.13) и HumidityRatio (0.3). Скорее всего эти признаки стоит исключить из модели, возможно они только ухудшат качество модели.
- Влажность и HumidityRatio очень сильно коррелируют между собой (0.96). Это неудивительно, ведь HumidityRatio величина производная от влажности. Поэтому из этих признаков в модели можно оставлять только один.
- Также можно сделать вывод, что выбирая из признаков влажность и HumidityRatio лучше выбрать HumidityRatio, потому что он сильнее коррелирован с целевым признаком. Если линейно зависимые признаки сильно коррелированы с целевым, то оставляют именно тот признак, который коррелирован с целевым сильнее. Но для этой пары признаков этот вывод нельзя считать надежным, потому что и 0.13 и 0.3 являются довольно малыми величинами.

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.

```
df.corr(method='pearson')
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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
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
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
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
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
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
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
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
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366081
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

```
df.corr(method='kendall')
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
CHAS	0.033948	-0.039419	0.075889	1.000000	0.056387	0.048080	0.055616	-0.065619	0.021739	-0.037655	-0.115694	-0.033271
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
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
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
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
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
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
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
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
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
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

```
df.corr(method='spearman')
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
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
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
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
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
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
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
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
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
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
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
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
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
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
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

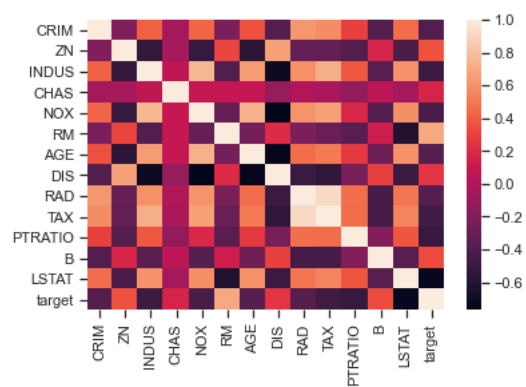
В случае большого количества признаков анализ числовой корреляционной матрицы становится неудобен.

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

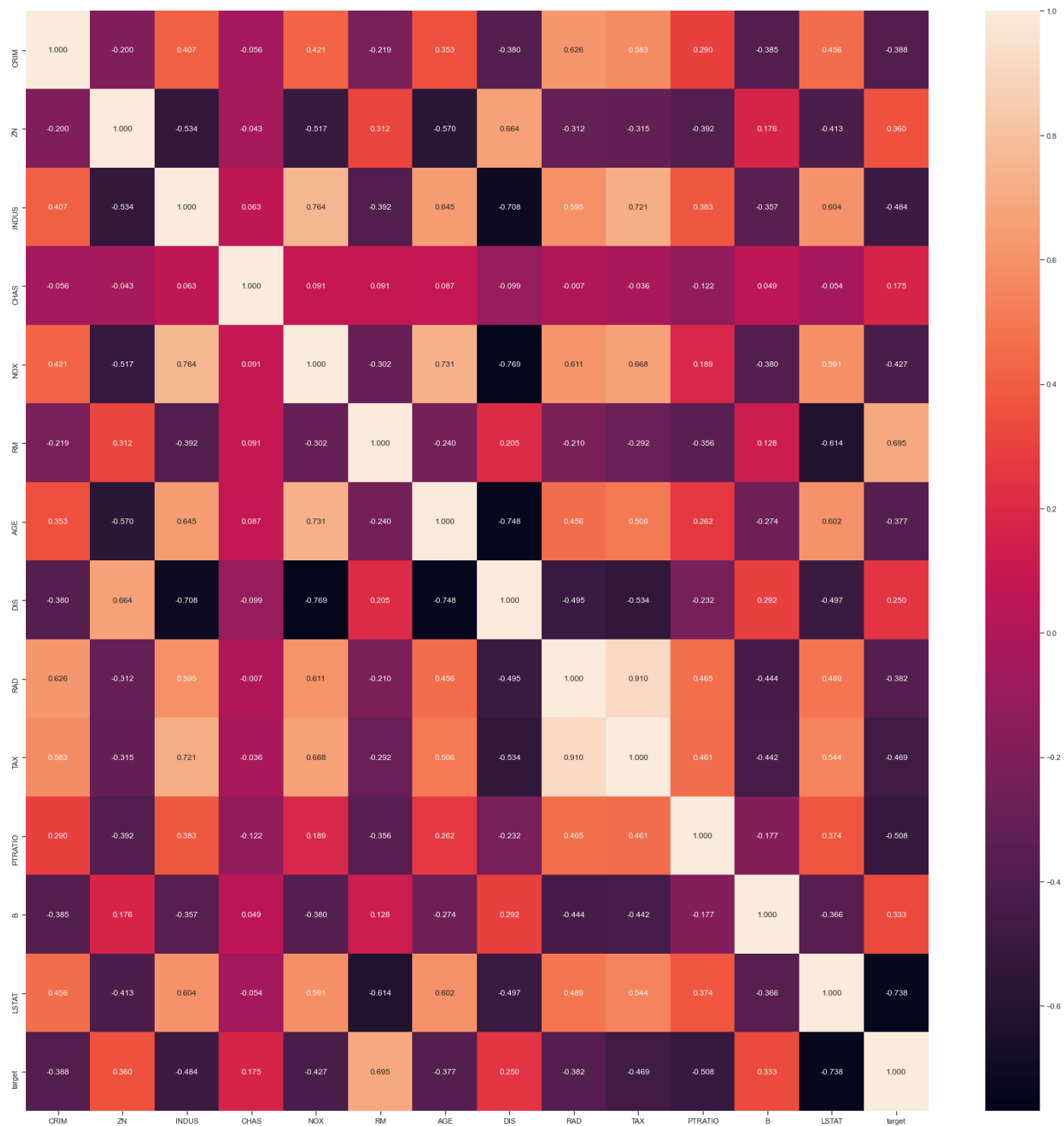
Используем метод heatmap библиотеки seaborn - <https://seaborn.pydata.org/generated/seaborn.heatmap.html>

```
sns.heatmap(df.corr())
```

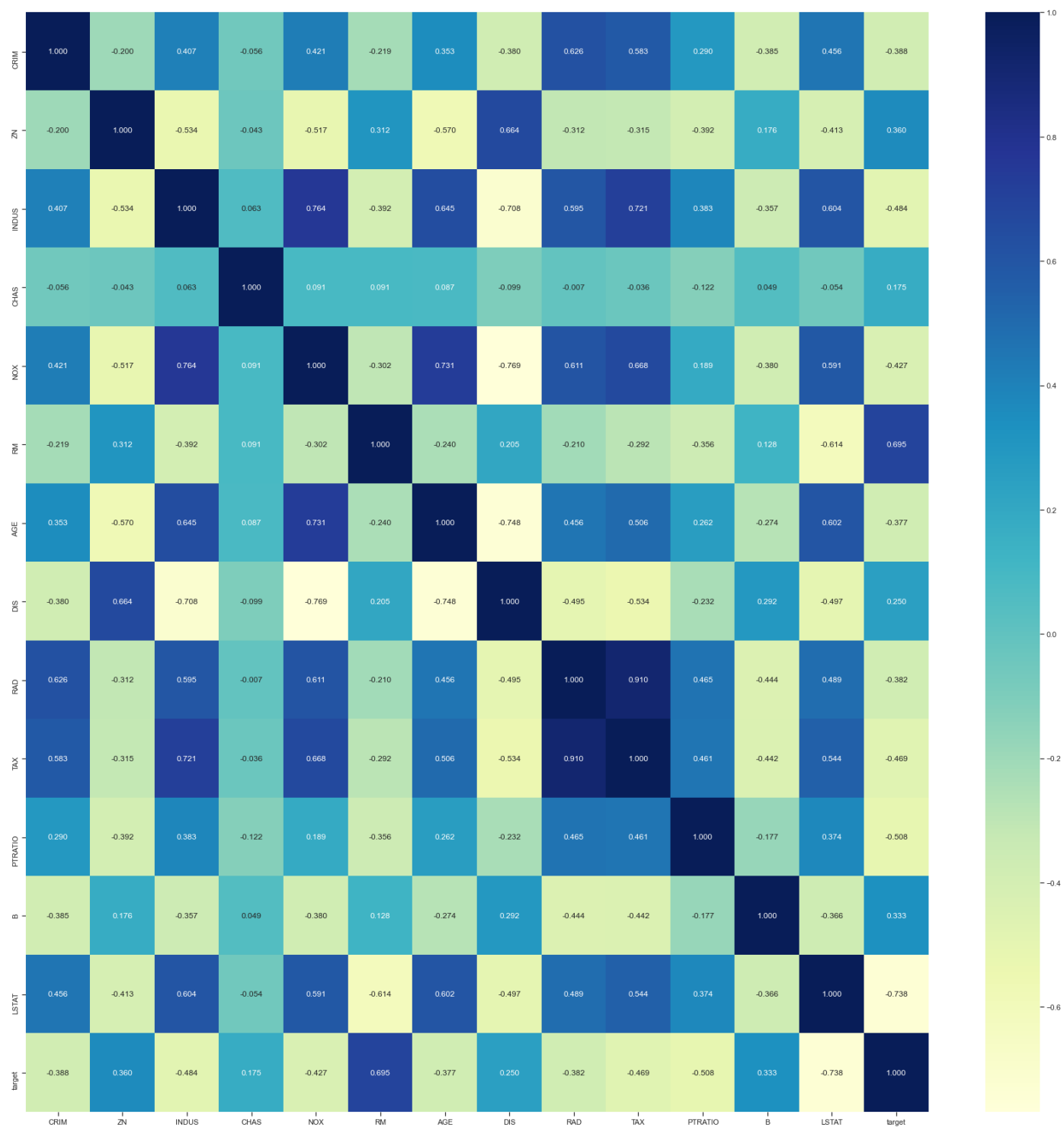
```
<matplotlib.axes._subplots.AxesSubplot at 0x2738baf6340>
```

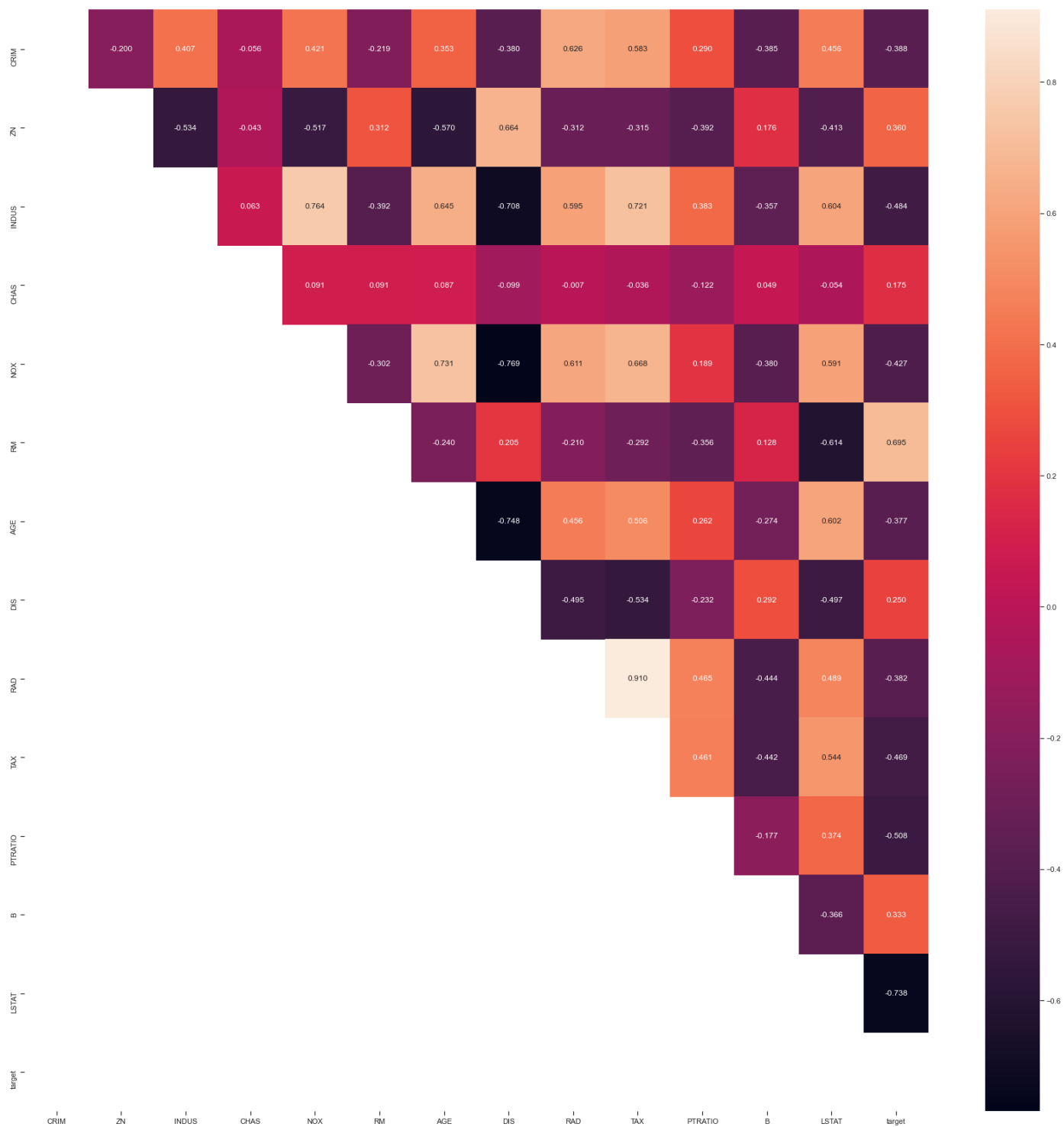
```
# Вывод значений в ячейках
fig, ax = plt.subplots(figsize=(30,30))
sns.heatmap(df.corr(), annot=True, fmt='.3f', ax=ax)
plt.show()
```



```
# Изменение цветовой гаммы
fg, ax = plt.subplots(figsize=(30,30))
sns.heatmap(df.corr(), cmap='YlGnBu', annot=True, fmt='.3f', ax=ax)
plt.show()
```



```
# Треугольный вариант матрицы
mask = np.zeros_like(df.corr(), dtype=np.bool)
# чтобы оставить нижнюю часть матрицы
# mask[np.triu_indices_from(mask)] = True
# чтобы оставить верхнюю часть матрицы
fg, ax = plt.subplots(figsize=(30,30))
mask[np.tril_indices_from(mask)] = True
sns.heatmap(df.corr(), mask=mask, annot=True, fmt='.3f', ax=ax)
plt.show()
```



```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(60,20))
sns.heatmap(df.corr(method='pearson'), ax=ax[0], annot=True, fmt='.2f')
sns.heatmap(df.corr(method='kendall'), ax=ax[1], annot=True, fmt='.2f')
sns.heatmap(df.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')
```

