Разведочный анализ данных. Исследование и визуализация данных.

1) Текстовое описание набора данных

В качестве набора данных был выбран известный датасет бостонского жилья.

Описание колонок:

- **crim** - per capita crime rate by town.
- **zn** - proportion of residential land zoned for lots over 25,000 sq.ft.
- **indus** - proportion of non-retail business acres per town.
- **chas** - Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- **nox** - nitrogen oxides concentration (parts per 10 million).
- **rm** - average number of rooms per dwelling.
- **age** - proportion of owner-occupied units built prior to 1940.
- **dis** - weighted mean of distances to five Boston employment centres.
- **rad** - index of accessibility to radial highways.
- **tax** - full-value property-tax rate per $10,000.
- **ptratio** - pupil-teacher ratio by town.
- **black** - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town.
- **lstat** - lower status of the population (percent).
- **medv** - median value of owner-occupied homes in $1000s.

Переменная medv является целевой.

Импорт библиотек

In [ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Загрузка данных

In [ ]:
```
data_path = 'data/Boston.csv'
df = pd.read_csv(data_path)
```

2) Основные характеристики датасета

Первые пять строчек

In [ ]:
```
df.head()
```

Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	I
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	39
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	39
2	3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	39
3	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	39
4	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	39

Определим размер датасета

In [ ]:
```
df.shape
```

Out[ ]:
```
(506, 15)
```

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

```
Всего строк: 506
```

In [ ]:
```
columns = df.columns
print('Колонки: {}'.format(columns))
```

```
Колонки: Index(['Unnamed: 0', 'crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'a
ge', 'dis',
       'rad', 'tax', 'ptratio', 'black', 'lstat', 'medv'],
      dtype='object')
```

Список колонок с типами данных:

In [ ]:
```
df.dtypes
```

Out[ ]:
```
Unnamed: 0      int64
crim          float64
zn            float64
indus         float64
chas            int64
nox           float64
rm            float64
age           float64
dis           float64
rad             int64
tax             int64
ptratio       float64
black         float64
lstat         float64
medv          float64
dtype: object
```

Проверка наличия пустых значений

In [ ]:
```
cols_with_nulls = []
for col in df.columns:
```

```
    # Количество пустых значений - все значения заполнены
    temp_null_count = df[df[col].isnull()].shape[0]
    if temp_null_count: cols_with_nulls.append(col)
    print('{} - {}'.format(col, temp_null_count))

if len(cols_with_nulls):
    print('Есть пустые значения')
else:
    print('Пустых значений нет')
```

```
Unnamed: 0 - 0
crim - 0
zn - 0
indus - 0
chas - 0
nox - 0
rm - 0
age - 0
dis - 0
rad - 0
tax - 0
ptratio - 0
black - 0
lstat - 0
medv - 0
Пустых значений нет
```

Основные статистические характеристки набора данных

In [ ]:
```
df.describe()
```

Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000
mean	253.500000	3.613524	11.363636	11.136779	0.069170	0.554695	6.284
std	146.213884	8.601545	23.322453	6.860353	0.253994	0.115878	0.702
min	1.000000	0.006320	0.000000	0.460000	0.000000	0.385000	3.561
25%	127.250000	0.082045	0.000000	5.190000	0.000000	0.449000	5.885
50%	253.500000	0.256510	0.000000	9.690000	0.000000	0.538000	6.208
75%	379.750000	3.677083	12.500000	18.100000	0.000000	0.624000	6.623
max	506.000000	88.976200	100.000000	27.740000	1.000000	0.871000	8.780

Определим уникальные значения для целевого признака - medv

In [ ]:
```
df['medv'].unique()
```

Out[ ]:
```
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])
```

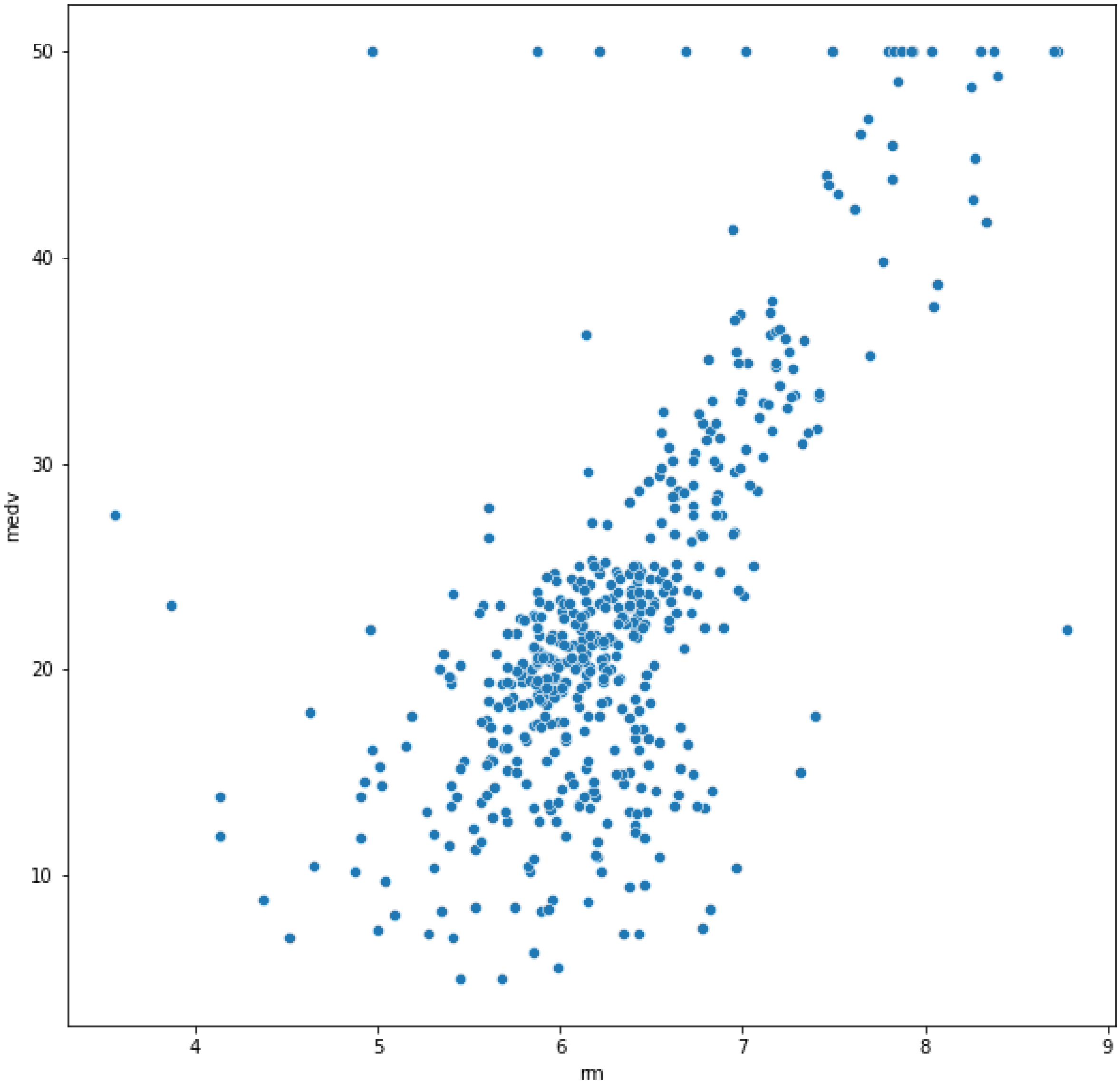
3) Визуальное исследование датасета

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

Диаграмма рассеяния

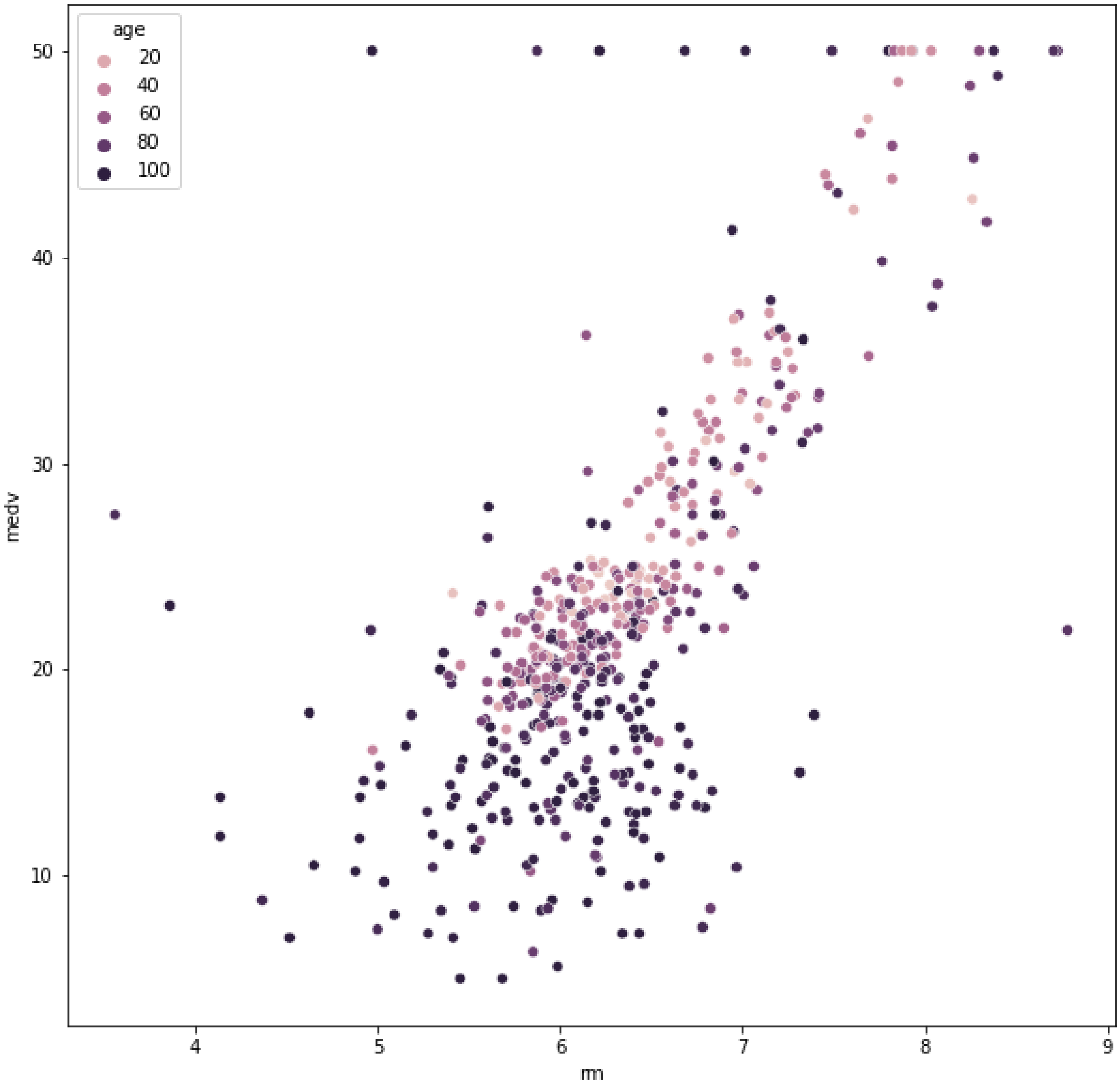
In [ ]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='rm', y='medv', data=df)
```

Out[ ]:
```
<AxesSubplot:xlabel='rm', ylabel='medv'>
```



In [ ]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(ax=ax, x='rm', y='medv', data=df, hue='age')
```

Out[ ]: <AxesSubplot:xlabel='rm', ylabel='medv'>

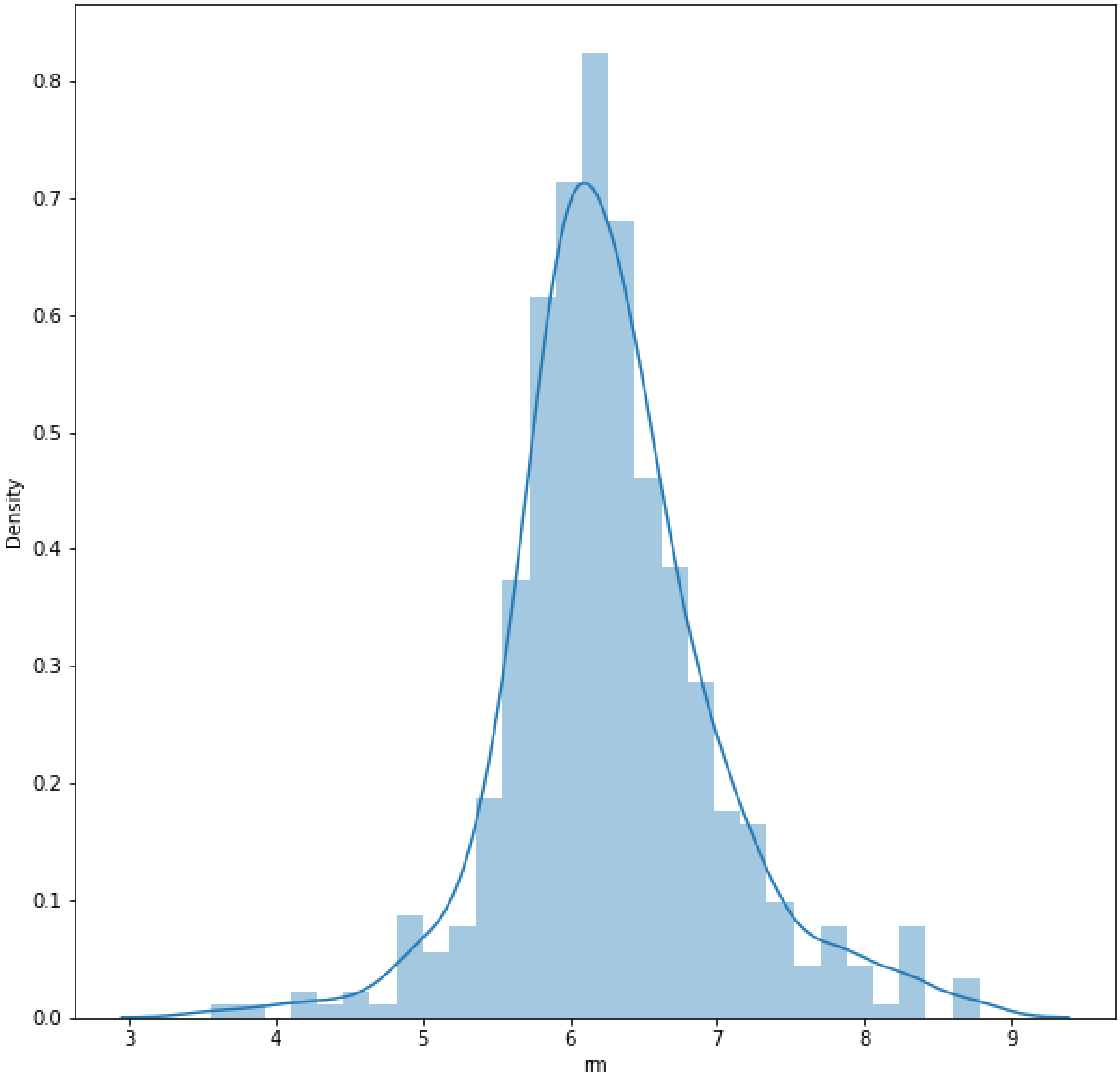


Гистограмма

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

```
/Users/feelsbadmans/Univer/bmstu-6-sem-tmo/.venv/lib/python3.8/site-package
s/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated f
unction and will be removed in a future version. Please adapt your code to u
se either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

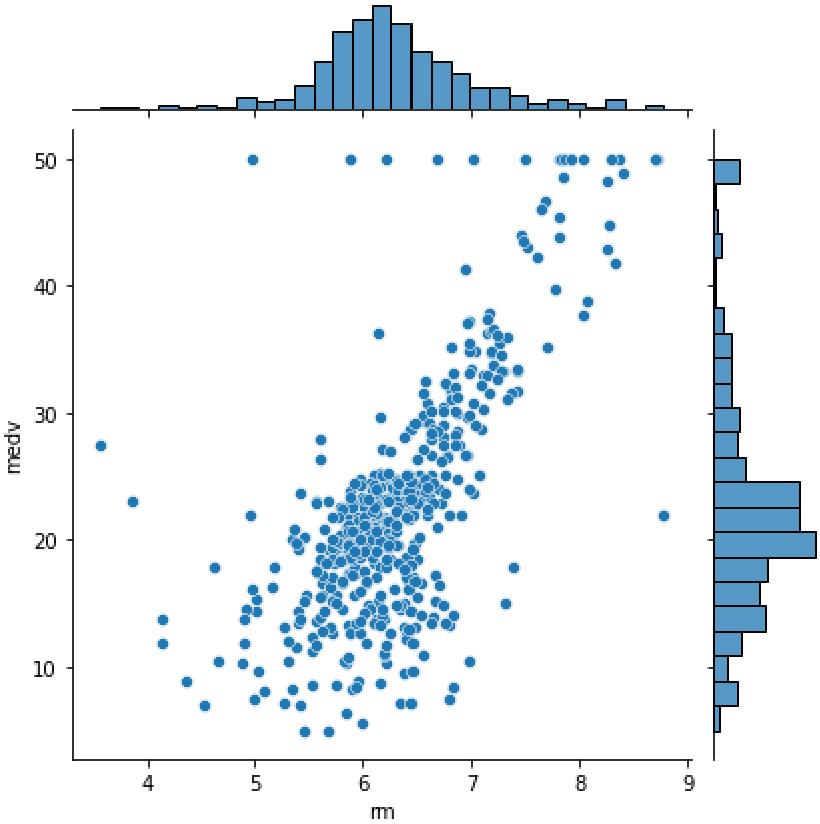
Out[ ]:
```
<AxesSubplot:xlabel='rm', ylabel='Density'>
```



Jointplot

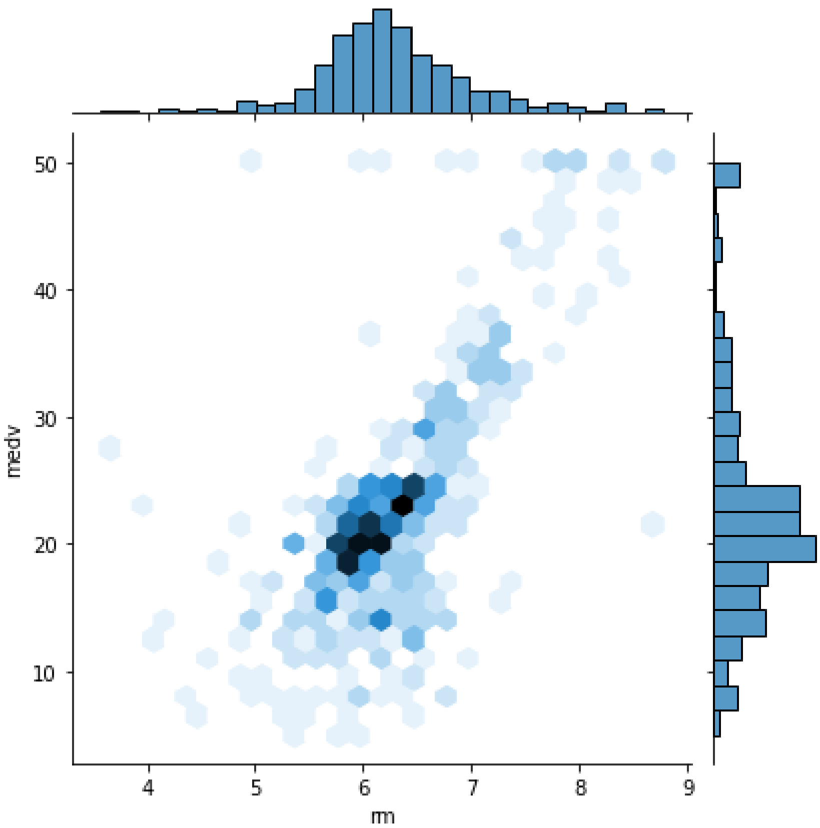
In [ ]:
```
sns.jointplot(x='rm', y='medv', data=df)
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x2b6cb4970>



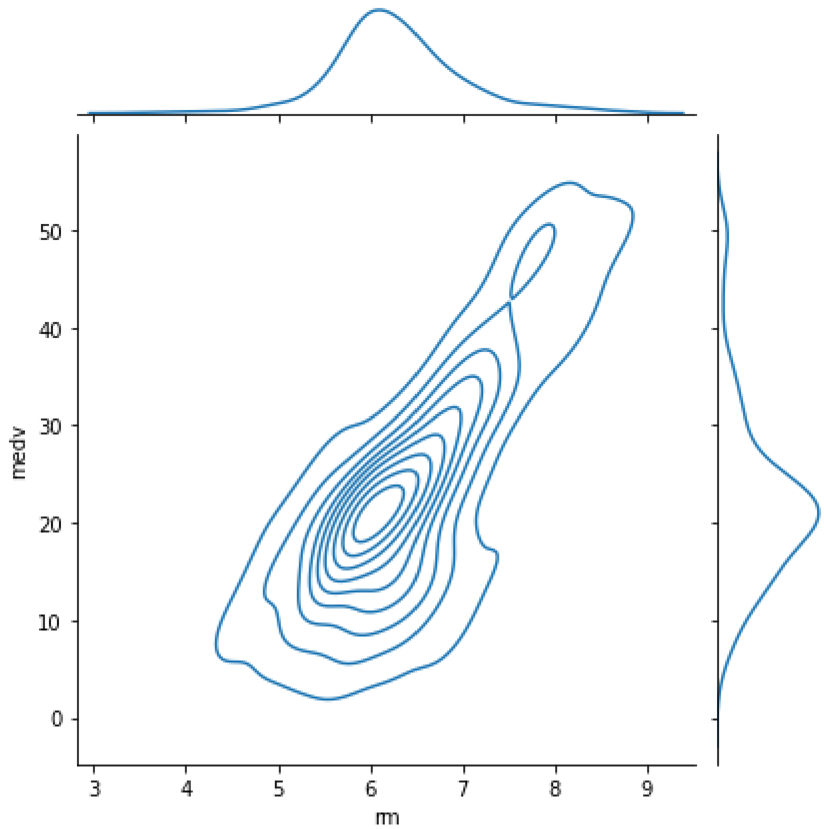
In [ ]:
```
sns.jointplot(x='rm', y='medv', data=df, kind='hex')
```

Out[ ]: <seaborn.axisgrid.JointGrid at 0x2b6e2ee50>



In [ ]:
```
sns.jointplot(x='rm', y='medv', data=df, kind='kde')
```

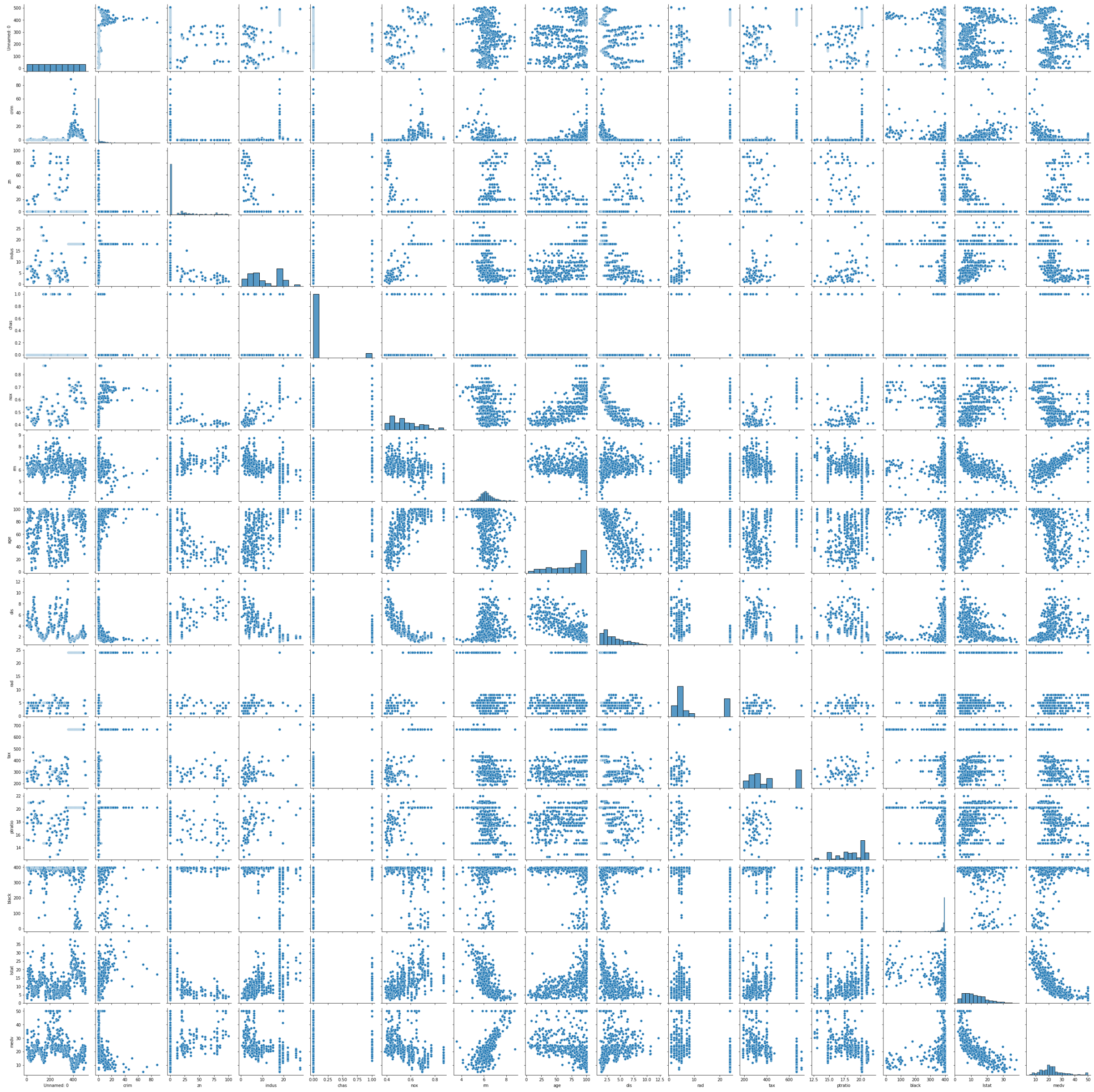
Out[ ]: <seaborn.axisgrid.JointGrid at 0x2b6f89c10>



"Парные диаграммы"

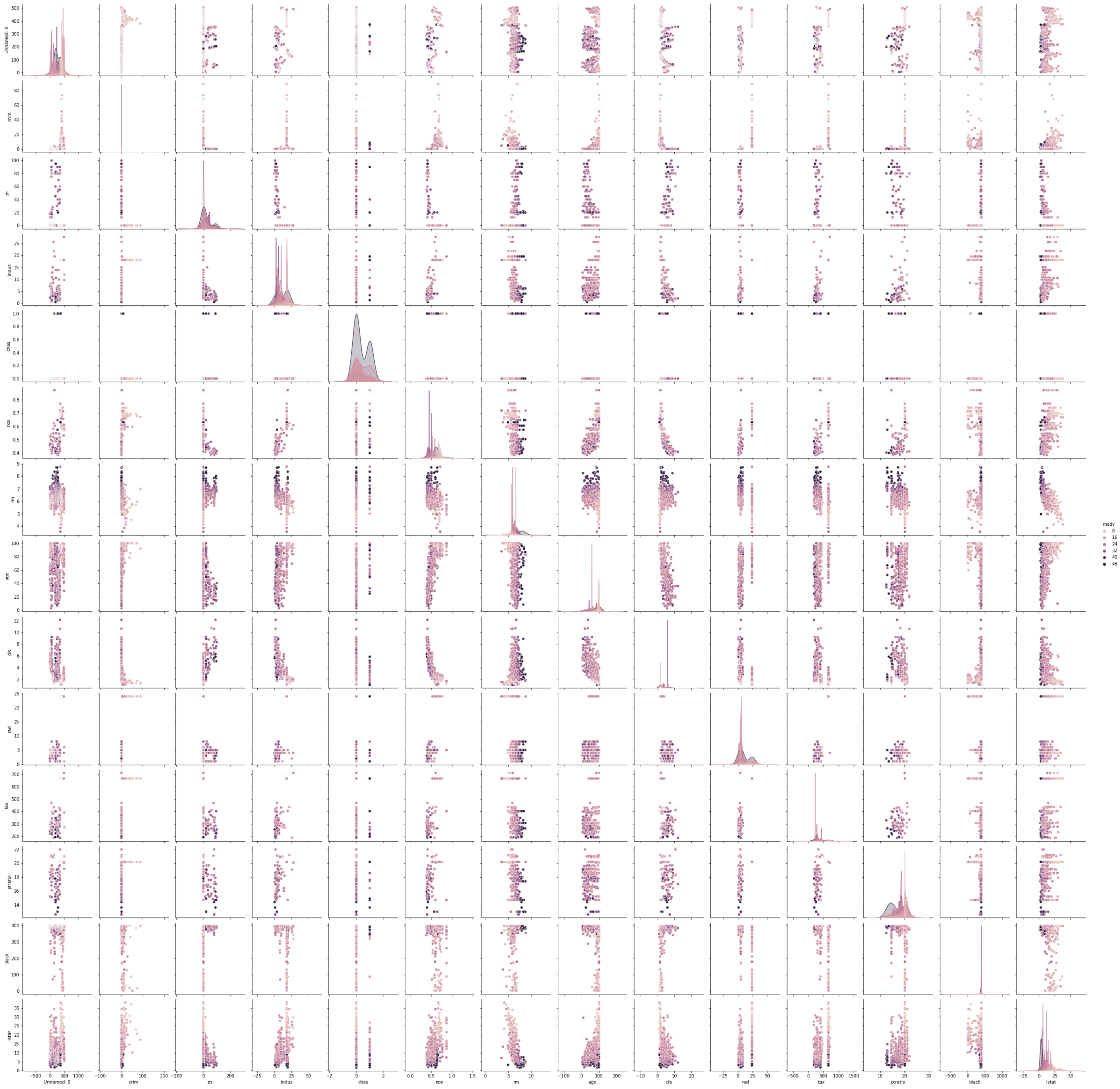
In [ ]:
```
sns.pairplot(df)
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x2b786e550>
```



In [ ]:
```
sns.pairplot(df, hue="medv")
```

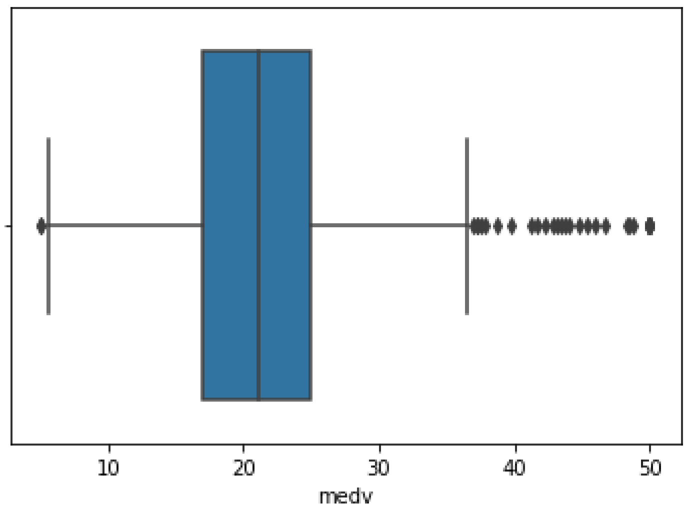
Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x2c0259940>
```



Ящик с усами

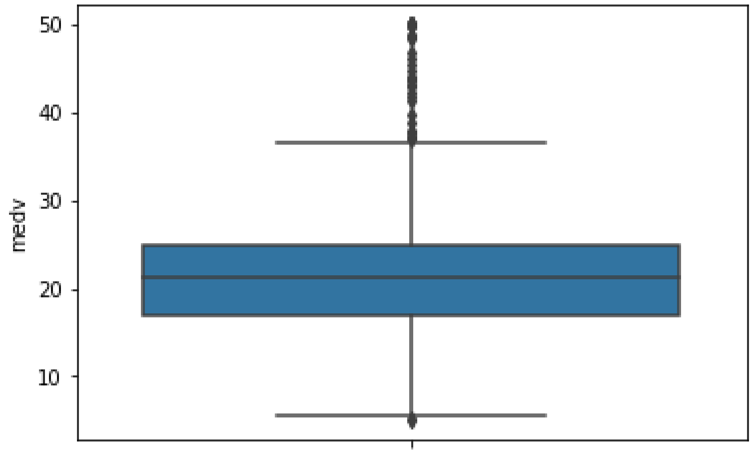
In [ ]:
```
sns.boxplot(x=df['medv'])
```

Out[ ]: <AxesSubplot:xlabel='medv'>



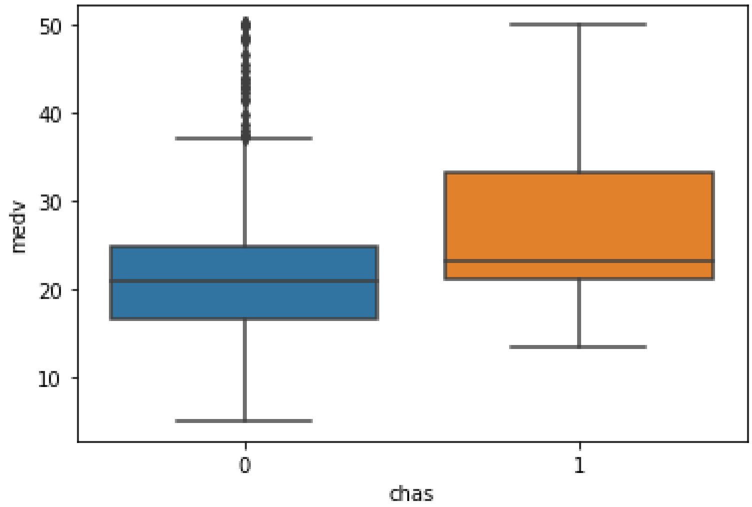
In [ ]:
```
sns.boxplot(y=df['medv'])
```

Out[ ]: <AxesSubplot:ylabel='medv'>



In [ ]:
```
sns.boxplot(x='chas', y='medv', data=df)
```

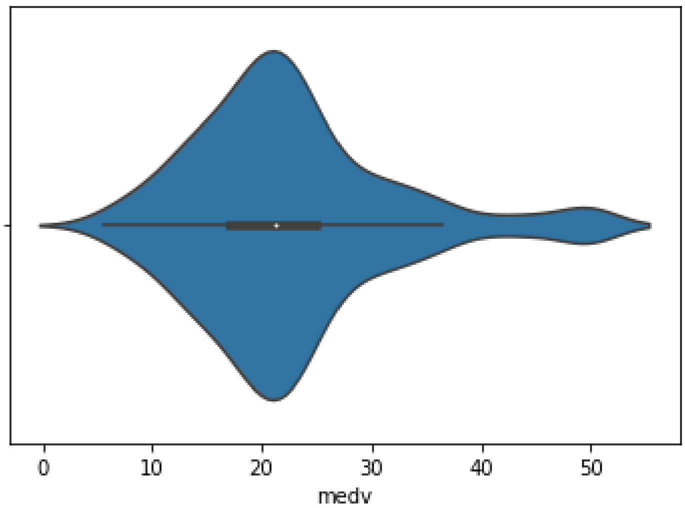
Out[ ]: <AxesSubplot:xlabel='chas', ylabel='medv'>



Violin plot

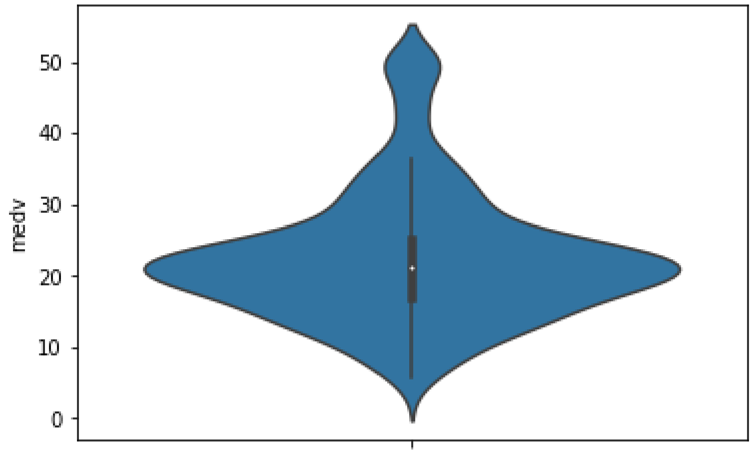
In [ ]:
```
sns.violinplot(x=df['medv'])
```

Out[ ]: <AxesSubplot:xlabel='medv'>



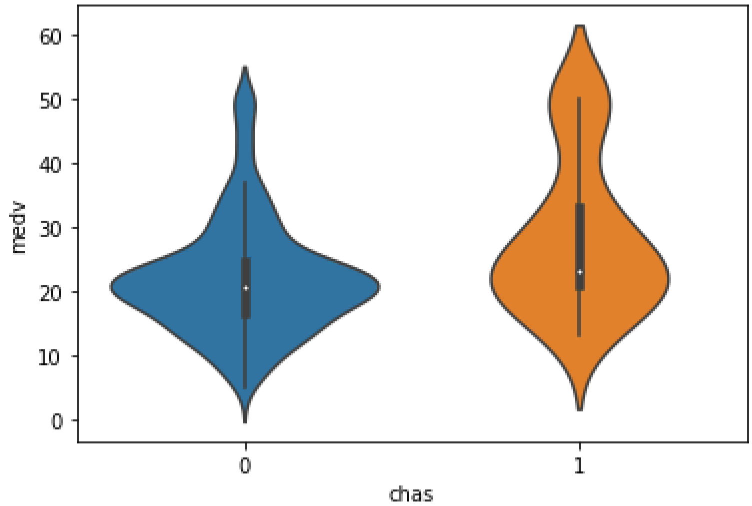
In [ ]:
```
sns.violinplot(y=df['medv'])
```

Out[ ]: <AxesSubplot:ylabel='medv'>



In [ ]:
```
sns.violinplot(x='chas', y='medv', data=df)
```

Out[ ]: `<AxesSubplot:xlabel='chas', ylabel='medv'>`



4) Информация о корреляции признаков

In [ ]:
```
df.corr()
```

Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm
Unnamed: 0	1.000000	0.407407	-0.103393	0.399439	-0.003759	0.398736	-0.079971
crim	0.407407	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247
zn	-0.103393	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991
indus	0.399439	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676
chas	-0.003759	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251
nox	0.398736	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188
rm	-0.079971	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000
age	0.203784	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265
dis	-0.302211	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246
rad	0.686002	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847
tax	0.666626	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048
ptratio	0.291074	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501
black	-0.295041	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069
lstat	0.258465	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808
medv	-0.226604	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360

In [ ]:

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

Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm
Unnamed: 0	1.000000	0.407407	-0.103393	0.399439	-0.003759	0.398736	-0.079971
crim	0.407407	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247
zn	-0.103393	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991
indus	0.399439	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676
chas	-0.003759	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251
nox	0.398736	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188
rm	-0.079971	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000
age	0.203784	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265
dis	-0.302211	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246
rad	0.686002	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847
tax	0.666626	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048
ptratio	0.291074	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501
black	-0.295041	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069
lstat	0.258465	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808
medv	-0.226604	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360

In [ ]:

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

Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm
Unnamed: 0	1.000000	0.287507	-0.119015	0.210280	-0.003072	0.270223	-0.025624
crim	0.287507	1.000000	-0.462057	0.521014	0.033948	0.603361	-0.211718
zn	-0.119015	-0.462057	1.000000	-0.535468	-0.039419	-0.511464	0.278134
indus	0.210280	0.521014	-0.535468	1.000000	0.075889	0.612030	-0.291318
chas	-0.003072	0.033948	-0.039419	0.075889	1.000000	0.056387	0.048080
nox	0.270223	0.603361	-0.511464	0.612030	0.056387	1.000000	-0.215633
rm	-0.025624	-0.211718	0.278134	-0.291318	0.048080	-0.215633	1.000000
age	0.131520	0.497297	-0.429389	0.489070	0.055616	0.589608	-0.187611
dis	-0.214680	-0.539878	0.478524	-0.565137	-0.065619	-0.683930	0.179801
rad	0.439464	0.563969	-0.234663	0.353967	0.021739	0.434828	-0.076569
tax	0.360426	0.544956	-0.289911	0.483228	-0.037655	0.453258	-0.190532
ptratio	0.222774	0.312768	-0.361607	0.336612	-0.115694	0.278678	-0.223194
black	-0.108716	-0.264378	0.128177	-0.192017	-0.033277	-0.202430	0.032951
lstat	0.159116	0.454837	-0.386818	0.465980	-0.041344	0.452005	-0.468231
medv	-0.170050	-0.403964	0.339989	-0.418430	0.115202	-0.394995	0.482829

In [ ]:

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

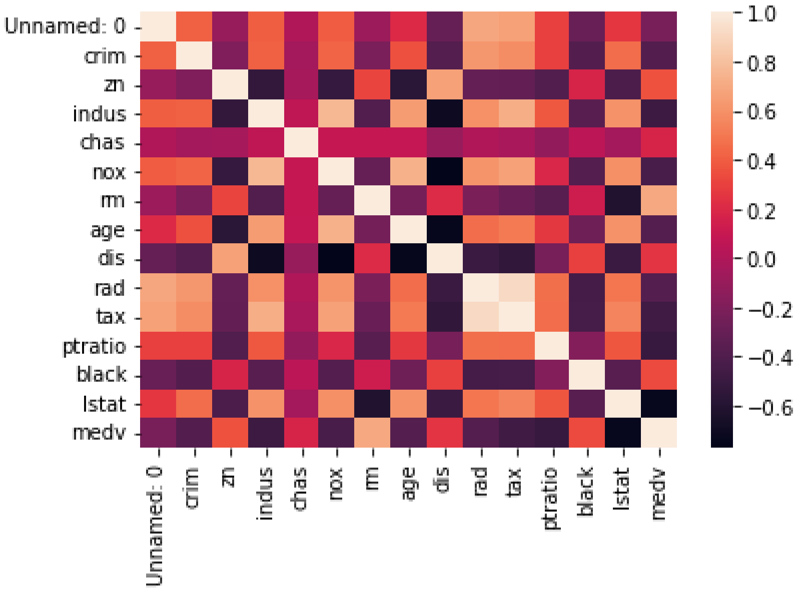
Out[ ]:

	Unnamed: 0	crim	zn	indus	chas	nox	rm
Unnamed: 0	1.000000	0.461037	-0.160505	0.324621	-0.003759	0.432492	-0.035641
crim	0.461037	1.000000	-0.571660	0.735524	0.041537	0.821465	-0.309116
zn	-0.160505	-0.571660	1.000000	-0.642811	-0.041937	-0.634828	0.361074
indus	0.324621	0.735524	-0.642811	1.000000	0.089841	0.791189	-0.415301
chas	-0.003759	0.041537	-0.041937	0.089841	1.000000	0.068426	0.058813
nox	0.432492	0.821465	-0.634828	0.791189	0.068426	1.000000	-0.310344
rm	-0.035641	-0.309116	0.361074	-0.415301	0.058813	-0.310344	1.000000
age	0.208323	0.704140	-0.544423	0.679487	0.067792	0.795153	-0.278082
dis	-0.373499	-0.744986	0.614627	-0.757080	-0.080248	-0.880015	0.263168
rad	0.588481	0.727807	-0.278767	0.455507	0.024579	0.586429	-0.107492
tax	0.536928	0.729045	-0.371394	0.664361	-0.044486	0.649527	-0.271898
ptratio	0.297897	0.465283	-0.448475	0.433710	-0.136065	0.391309	-0.312923
black	-0.154474	-0.360555	0.163135	-0.285840	-0.039810	-0.296662	0.053660
lstat	0.257542	0.634760	-0.490074	0.638747	-0.050575	0.636828	-0.640832
medv	-0.273633	-0.558891	0.438179	-0.578255	0.140612	-0.562609	0.633576

In [ ]:

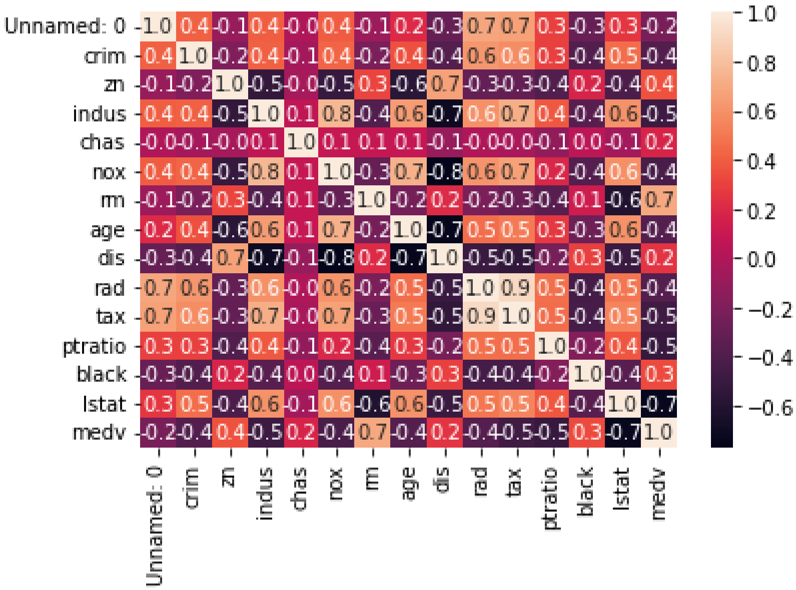
```
sns.heatmap(df.corr())
```

Out[ ]: <AxesSubplot:>



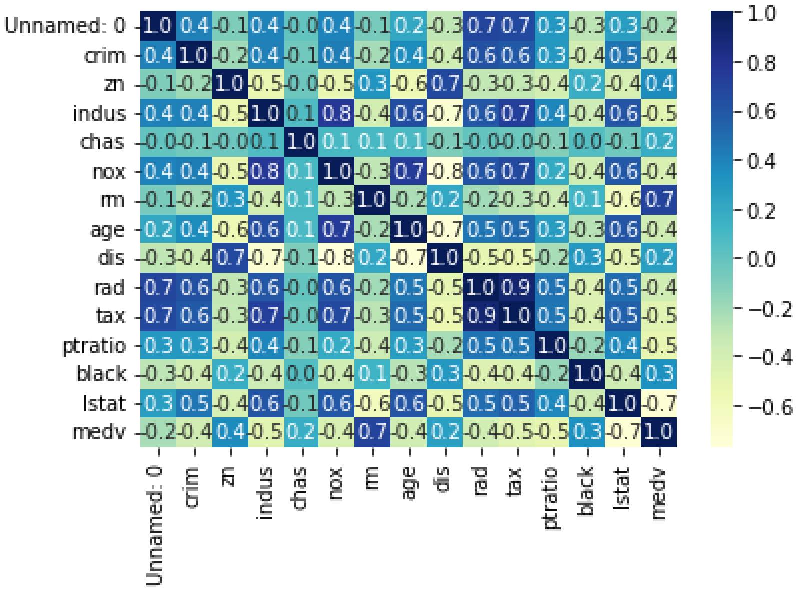
In [ ]:
```python
sns.heatmap(df.corr(), annot=True, fmt='.1f')
```

Out[ ]: <AxesSubplot:>



In [ ]:
```python
sns.heatmap(df.corr(), annot=True, fmt='.1f', cmap='YlGnBu')
```

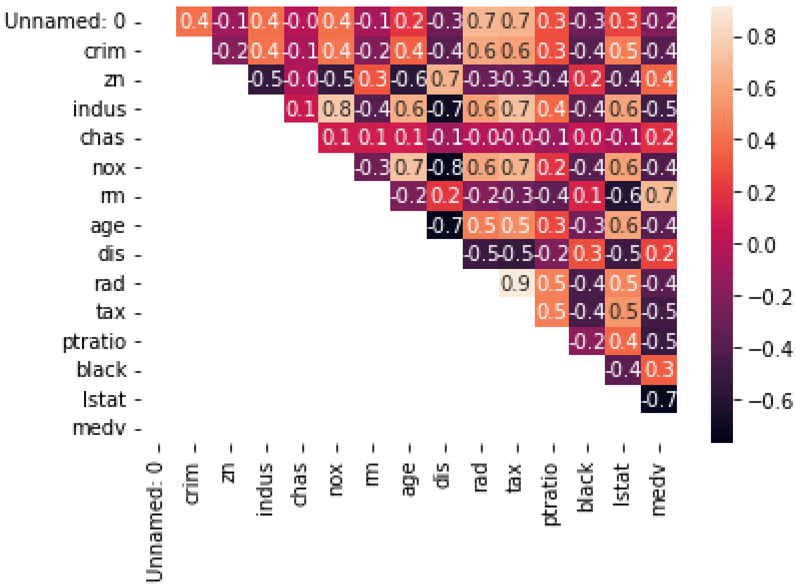
Out[ ]: <AxesSubplot:>



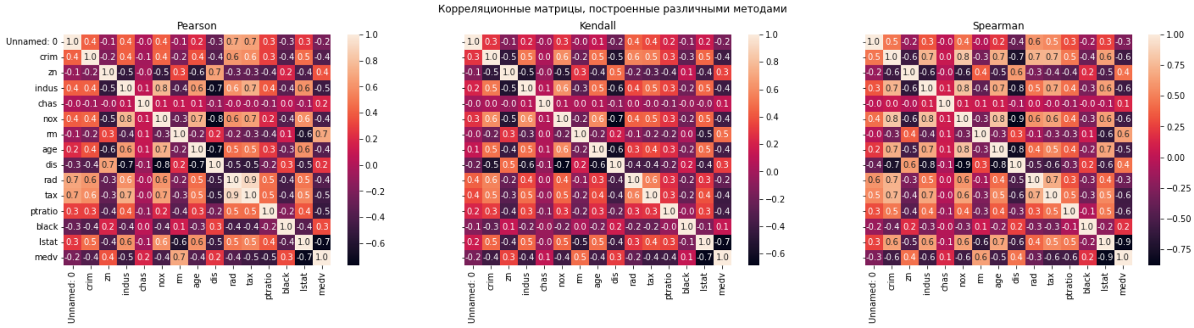
In [ ]:
```
mask = np.zeros_like(df.corr(), dtype=np.bool)
mask[np.tril_indices_from(mask)] = True
sns.heatmap(df.corr(), mask=mask, annot=True, fmt='.1f')
```

```
/var/folders/k8/94s33qgs47b0y9dc3zwr8t240000gn/T/ipykernel_62257/1732128229.py:1: 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(df.corr(), dtype=np.bool)
```

Out[ ]: <AxesSubplot:>



In [ ]:
```
fig, ax = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(24,5))
sns.heatmap(df.corr(method='pearson'), ax=ax[0], annot=True, fmt='.1f')
sns.heatmap(df.corr(method='kendall'), ax=ax[1], annot=True, fmt='.1f')
sns.heatmap(df.corr(method='spearman'), ax=ax[2], annot=True, fmt='.1f')
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')
```


Корреляционные матрицы, построенные различными методами
Pearson
Kendall
Spearman
Unnamed: 0
crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
medv