Project "Boston housing"

Stanin Vladislav

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm

from scipy.stats import ttest_ind, ttest_rel, mannwhitneyu, pearsonr
```

Data installation

In [2]:
```
data  = pd.read_csv('../data/BostonHousing.csv')
data
```

Out[2]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88	11.9

506 rows × 14 columns

little EDA

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

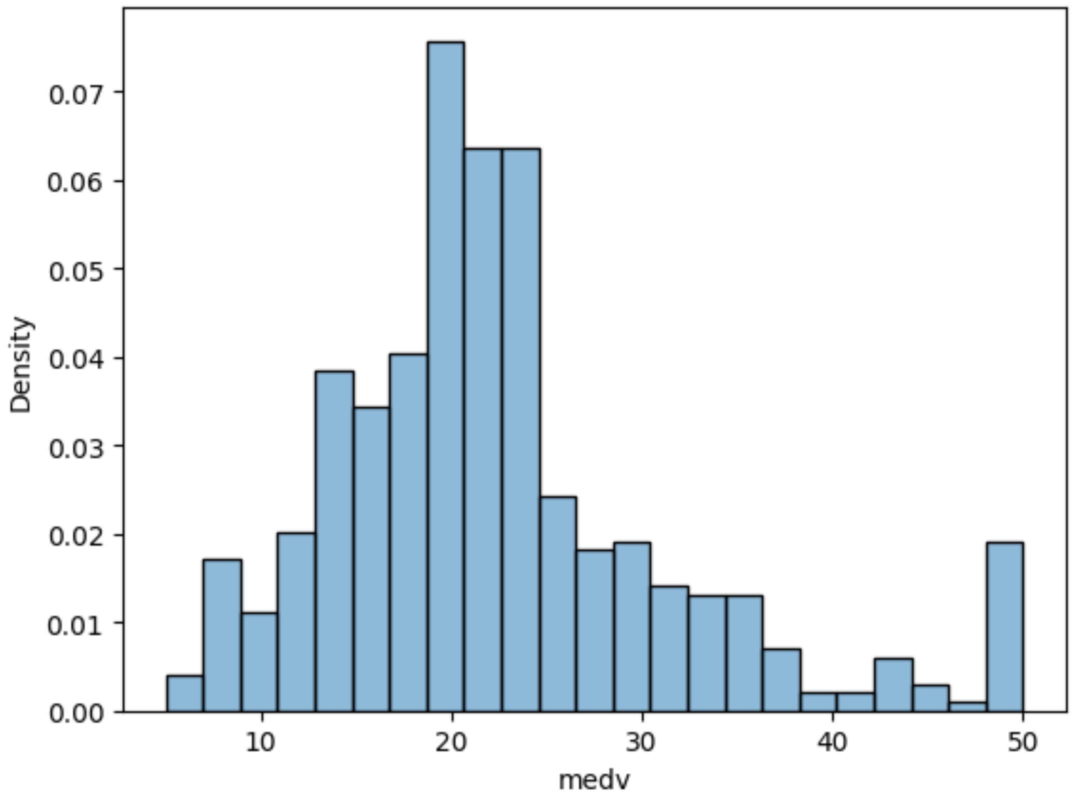
In [4]:
```
data.isna().sum()
```

Out[4]:
```
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
medv       0
dtype: int64
```

No problems with data

Distribution of target variable (*medv*):

In [5]:
```python
sns.histplot(data[['medv']], stat='density', color='blue', legend=False);
plt.xlabel('medv');
```



Looks like normal

In [6]:
```python
predictors = data.iloc[:,:-1]
medv = data.iloc[:,-1]
```

Standartization

In [7]:
```python
means = predictors.mean(axis=0)
stds = predictors.std(axis=0)

scaled_preds = (predictors - means) / stds

scaled_preds.head()
```

Out[7]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	
0	-0.419367	0.284548	-1.286636	-0.272329	-0.144075	0.413263	-0.119895	0.140075	-0.981871	-0.665949	-1.457558	0.440616	-1.074
1	-0.416927	-0.487240	-0.592794	-0.272329	-0.739530	0.194082	0.366803	0.556609	-0.867024	-0.986353	-0.302794	0.440616	-0.49
2	-0.416929	-0.487240	-0.592794	-0.272329	-0.739530	1.281446	-0.265549	0.556609	-0.867024	-0.986353	-0.302794	0.396035	-1.20
3	-0.416338	-0.487240	-1.305586	-0.272329	-0.834458	1.015298	-0.809088	1.076671	-0.752178	-1.105022	0.112920	0.415751	-1.36
4	-0.412074	-0.487240	-1.305586	-0.272329	-0.834458	1.227362	-0.510674	1.076671	-0.752178	-1.105022	0.112920	0.440616	-1.02

First linear model

In [8]:
```python
X = sm.add_constant(scaled_preds)
model_scaled = sm.OLS(medv, X)
results_scaled = model_scaled.fit()

print(results_scaled.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.734
Method:                 Least Squares   F-statistic:                     108.1
Date:                Tue, 13 Dec 2022   Prob (F-statistic):          6.72e-135
Time:                        14:51:45   Log-Likelihood:                -1498.8
No. Observations:                 506   AIC:                             3026.
Df Residuals:                     492   BIC:                             3085.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         22.5328      0.211    106.814      0.000      22.118      22.947
crim          -0.9291      0.283     -3.287      0.001      -1.484      -0.374
zn             1.0826      0.320      3.382      0.001       0.454       1.712
indus          0.1410      0.422      0.334      0.738      -0.688       0.970
chas           0.6824      0.219      3.118      0.002       0.252       1.112
nox           -2.0588      0.443     -4.651      0.000      -2.928      -1.189
rm             2.6769      0.294      9.116      0.000       2.100       3.254
age            0.0195      0.372      0.052      0.958      -0.711       0.750
dis           -3.1071      0.420     -7.398      0.000      -3.932      -2.282
rad            2.6649      0.578      4.613      0.000       1.530       3.800
tax           -2.0788      0.634     -3.280      0.001      -3.324      -0.834
ptratio       -2.0626      0.283     -7.283      0.000      -2.619      -1.506
b              0.8501      0.245      3.467      0.001       0.368       1.332
lstat         -3.7473      0.362    -10.347      0.000      -4.459      -3.036
==============================================================================
Omnibus:                      178.041   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              783.126
Skew:                           1.521   Prob(JB):                    8.84e-171
Kurtosis:                       8.281   Cond. No.                         9.82
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The *age* and *indus* predictors not significantly predicts the *medv*

Checking model without these predictors:

In [9]:
```
model_new = sm.OLS(medv, X.drop(columns=["indus", "age"]))
results_new = model_new.fit()

print(results_new.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.741
Model:                            OLS   Adj. R-squared:                  0.735
Method:                 Least Squares   F-statistic:                     128.2
Date:                Tue, 13 Dec 2022   Prob (F-statistic):          5.54e-137
Time:                        14:51:45   Log-Likelihood:                -1498.9
No. Observations:                 506   AIC:                             3022.
Df Residuals:                     494   BIC:                             3072.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         22.5328      0.211    107.018      0.000      22.119      22.946
crim          -0.9325      0.282     -3.307      0.001      -1.486      -0.379
zn             1.0692      0.315      3.390      0.001       0.450       1.689
chas           0.6905      0.217      3.183      0.002       0.264       1.117
nox           -2.0135      0.410     -4.915      0.000      -2.818      -1.209
rm             2.6711      0.285      9.356      0.000       2.110       3.232
dis           -3.1432      0.391     -8.037      0.000      -3.912      -2.375
rad            2.6088      0.552      4.726      0.000       1.524       3.693
tax           -1.9850      0.568     -3.493      0.001      -3.102      -0.868
ptratio       -2.0492      0.279     -7.334      0.000      -2.598      -1.500
b              0.8482      0.244      3.475      0.001       0.369       1.328
lstat         -3.7316      0.339    -11.019      0.000      -4.397      -3.066
==============================================================================
Omnibus:                      178.430   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              787.785
Skew:                           1.523   Prob(JB):                    8.60e-172
Kurtosis:                       8.300   Cond. No.                         7.90
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Model did not changed significantly arfter removing those predictors.

Nevertheless, F-statistics and R-adj were increased (and p-val decreased).

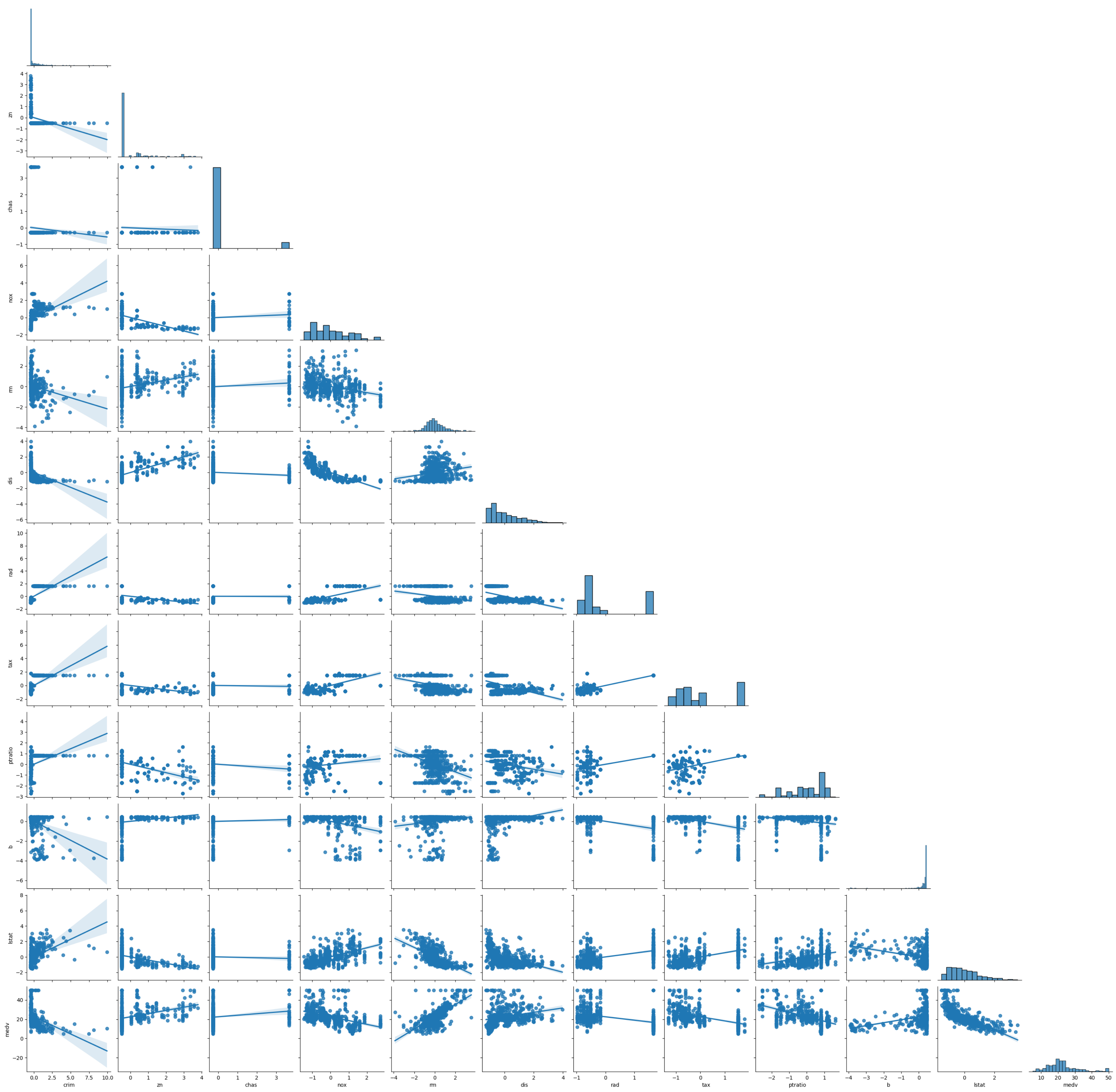
In the rest of report i will name these paramters as **Statistics** (If they become better, I will say that Statistics *increased*)

In [10]:
```
X_new = X.drop(columns=["indus", "age"])
```

Checking model

Linear relationship

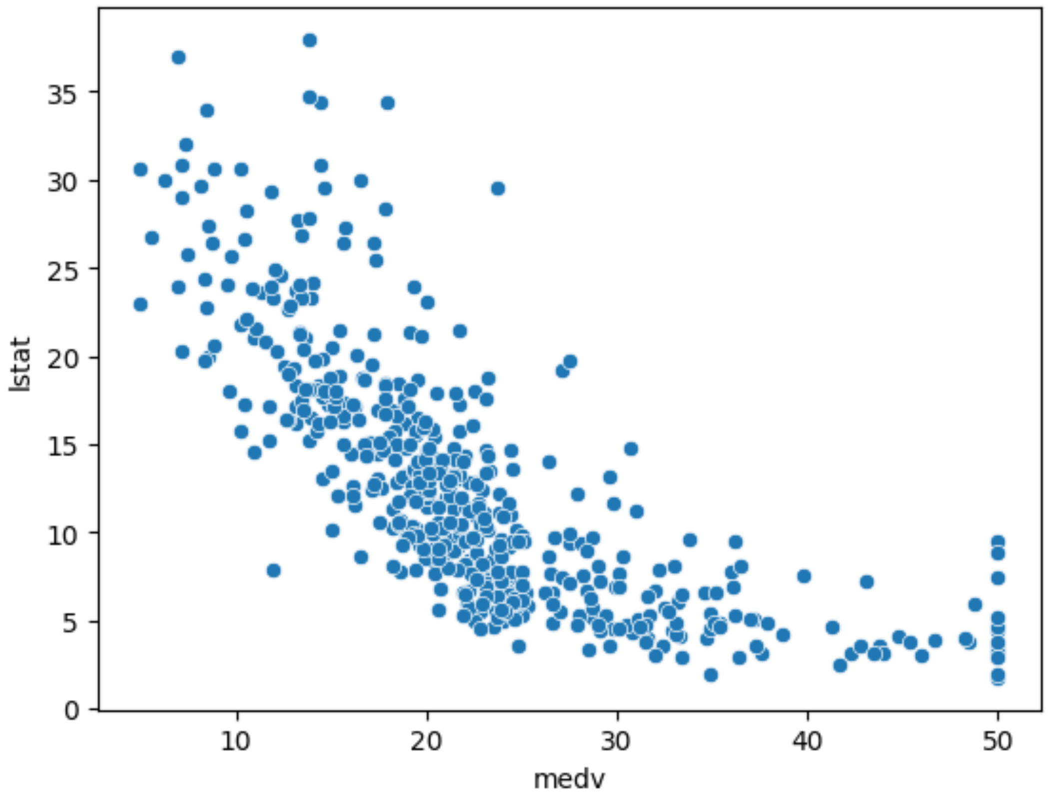
In [11]:
```
fig = sns.pairplot(pd.concat([scaled_preds.drop(columns=["indus", "age"]), medv], axis=1), kind="reg", corner=True);
fig.savefig("pairplot.png");
```



Most of predictors have linear relationship with *medv*, but *lstat* which have looking like exponental relationship

Let's check:

In [12]:
```
sns.scatterplot(y=predictors.lstat,
                x=medv);
```

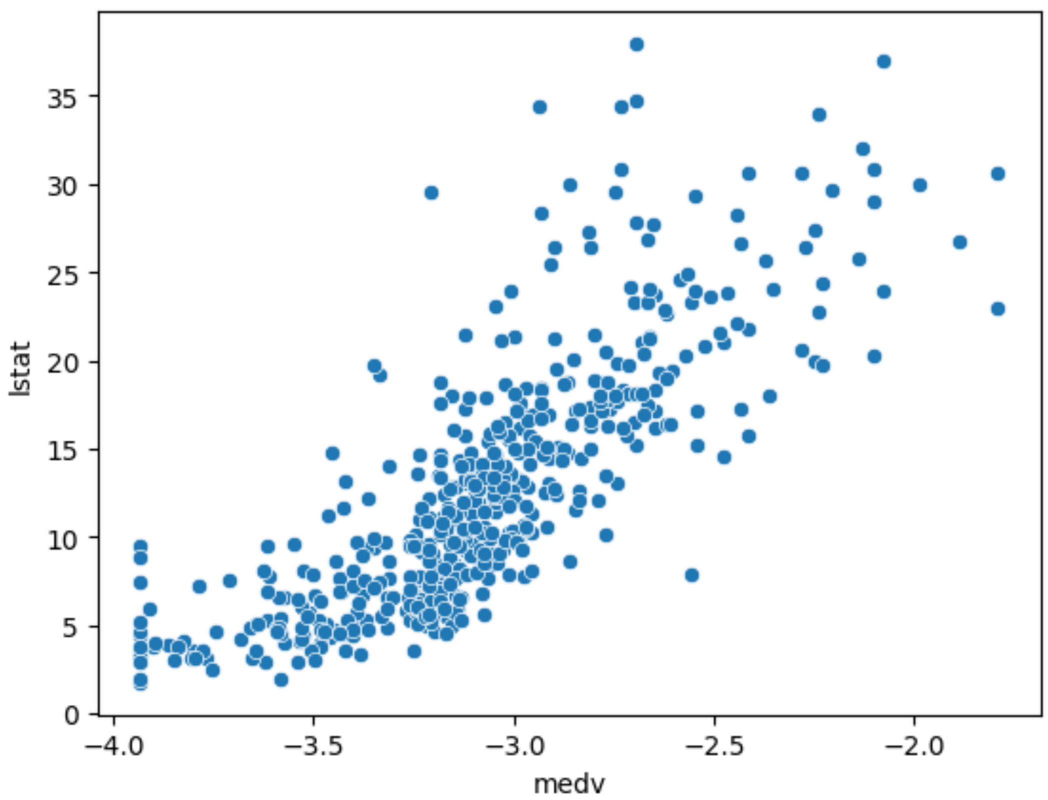



```
In [13]: print(f"medv and lstat correlated with Pearson\'s r value {pearsonr(y=predictors.lstat, x=medv).statistic} and pvalu
```

```
medv and lstat correlated with Pearson's r value -0.737662726174015 and pvalue 5.081103394386392e-88
```

Checking relationship between *lstat* and *-ln(medv)*

```
In [14]: sns.scatterplot(y=predictors.lstat,
                         x=-np.log(medv+1));
```




```
In [15]: print(f'ln(medv) and lstat correlated with Pearson\'s r value {pearsonr(y=predictors.lstat, x=np.log(medv+1)).statis
```

```
ln(medv) and lstat correlated with Pearson's r value -0.8043 and pvalue 5.230310856128727e-116
```

Since Pearson's r not changed significantly, we can leave it
However,
Additional check of model without *lstat* predictor:

```
In [16]: model_wo_lstat = sm.OLS(medv, X_new.drop(columns=["lstat"]))
         results_wo_lstat = model_wo_lstat.fit()

         print(results_wo_lstat.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.677
Model:                            OLS   Adj. R-squared:                  0.670
Method:                 Least Squares   F-statistic:                     103.7
Date:                Tue, 13 Dec 2022   Prob (F-statistic):          1.33e-114
Time:                        14:52:35   Log-Likelihood:                -1554.5
No. Observations:                 506   AIC:                             3131.
Df Residuals:                     495   BIC:                             3177.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         22.5328      0.235     95.979      0.000      22.072      22.994
crim          -1.4336      0.310     -4.621      0.000      -2.043      -0.824
zn             1.0549      0.352      3.000      0.003       0.364       1.746
chas           0.7784      0.242      3.220      0.001       0.303       1.253
nox           -2.9458      0.447     -6.591      0.000      -3.824      -2.068
rm             4.2960      0.273     15.761      0.000       3.760       4.831
dis           -2.7631      0.434     -6.361      0.000      -3.617      -1.910
rad            2.6869      0.616      4.365      0.000       1.478       3.896
tax           -2.1965      0.633     -3.468      0.001      -3.441      -0.952
ptratio       -2.3219      0.310     -7.482      0.000      -2.932      -1.712
b              1.2392      0.269      4.602      0.000       0.710       1.768
==============================================================================
Omnibus:                      247.217   Durbin-Watson:                   0.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             2087.021
Skew:                           1.949   Prob(JB):                         0.00
Kurtosis:                      12.154   Cond. No.                         7.46
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

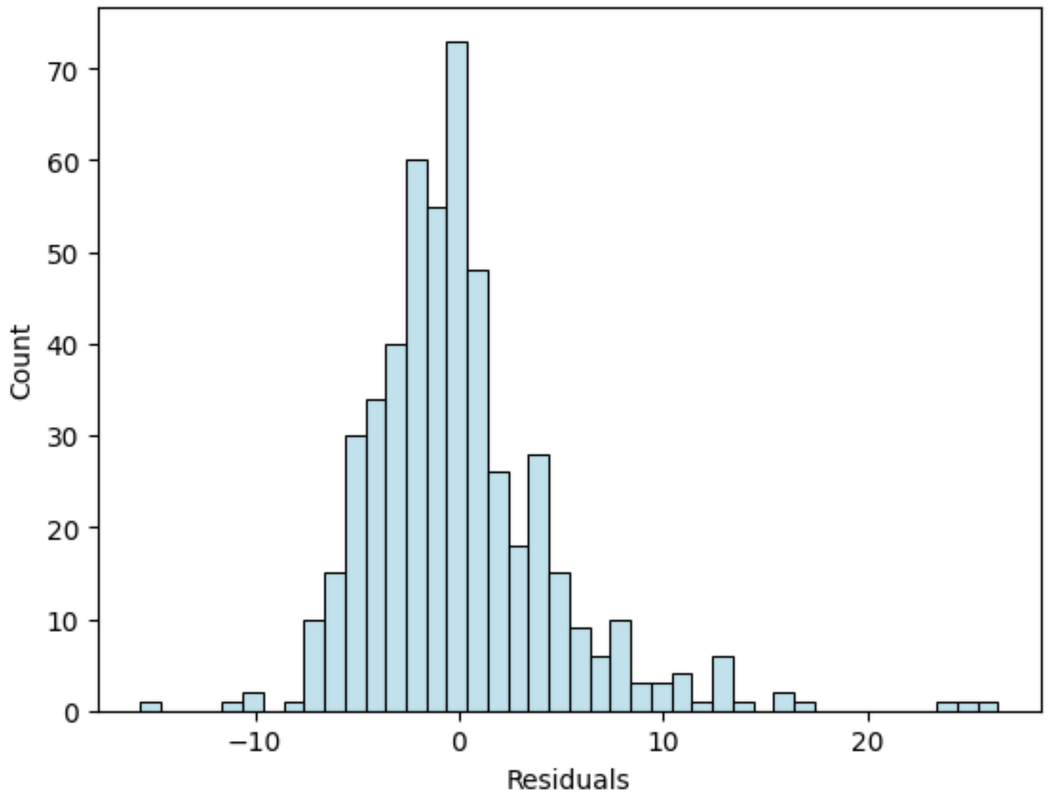
Statistics of model decresed, therefore we leave *lstat* predictor in model

Checking if distribution of residuals is normal

In [17]:
```python
prediction = results_new.get_prediction(X_new)
medv_predicted = prediction.predicted_mean
```

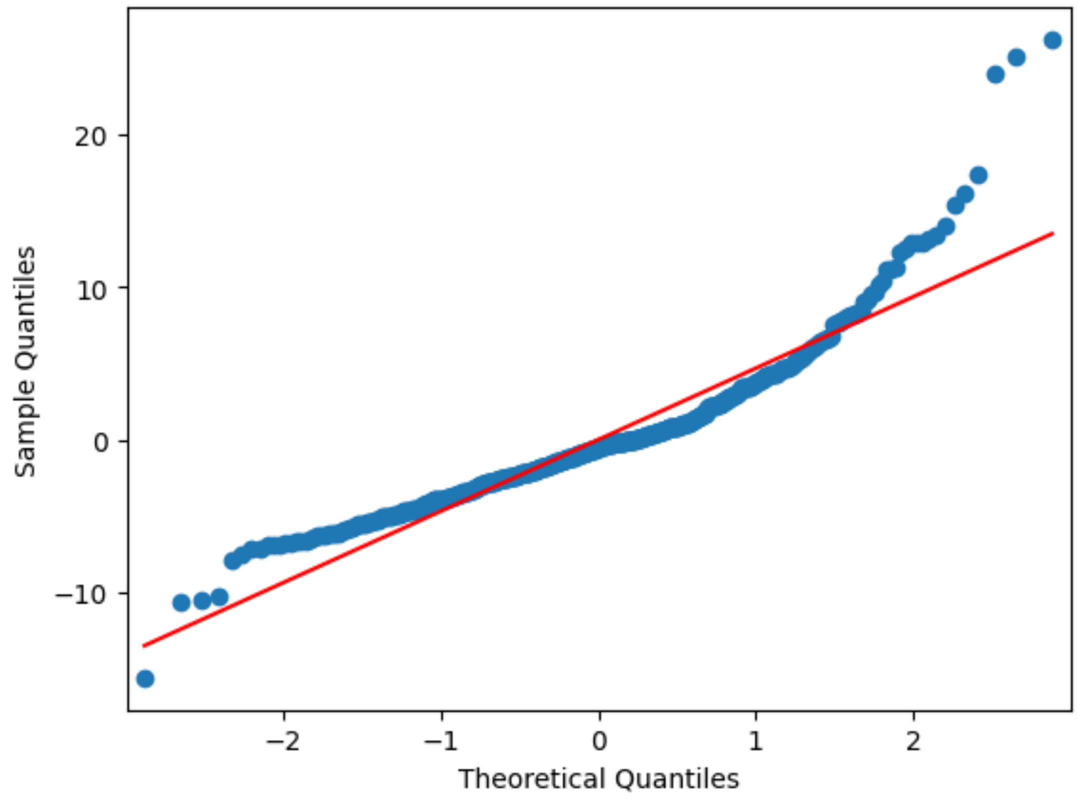
In [18]:
```python
residuals = medv - medv_predicted
```

In [19]:
```python
sns.histplot(residuals, color='lightblue', binwidth=1);
plt.xlabel('Residuals');
```



Seems like normal...

In [20]:
```python
sm.qqplot(residuals, line='s');
```



Still lloks like normal, with some deviations (richest houses)
but let's check these deviations:

Checking devations

(Calculating Cook's distances)

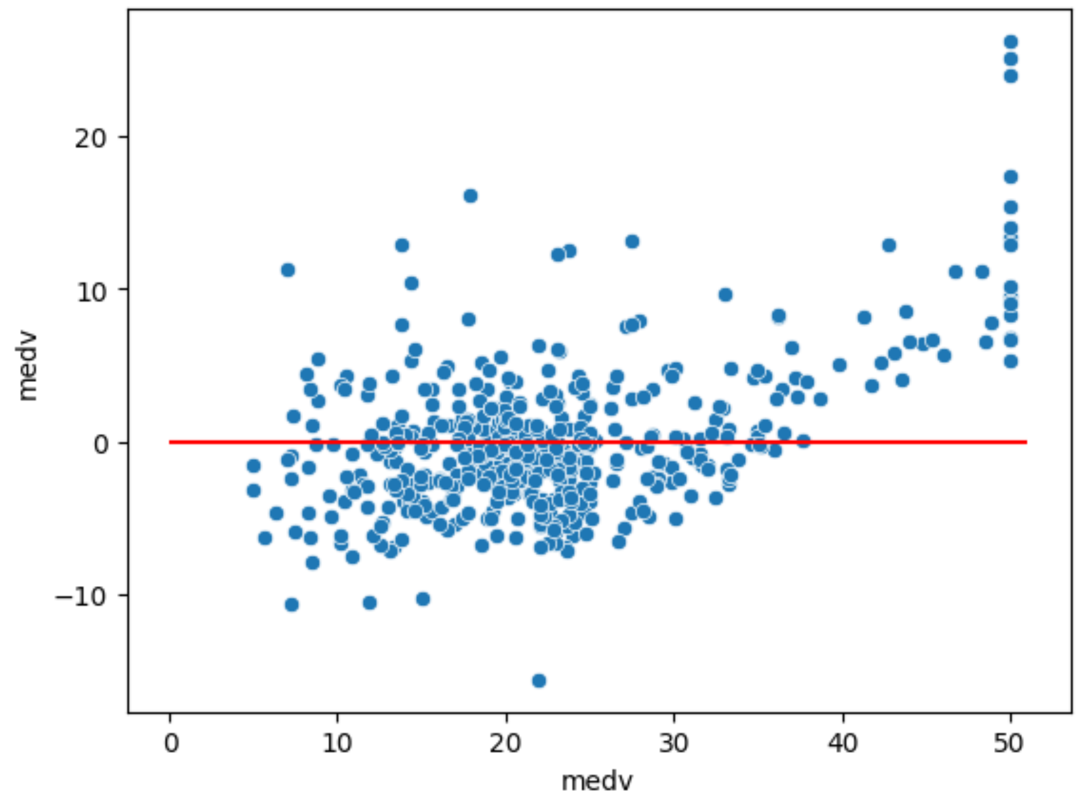
In [21]:
```
influence = results_new.get_influence()
cooks = influence.cooks_distance
n_deviations = (cooks[1] < 0.05).sum()
print(f'There are {n_deviations} significant deviations')
```

```
There are 0 significant deviations
```

There are no deviations!

Checking homoscedacity

In [22]:
```
sns.scatterplot(x = medv, y = residuals);
plt.hlines(0,0, 51, color = 'red');
```



Only some very expensive houses deviate. Nevertheless, we can consider that the model has homoscedasticity

Checking VIF

In [23]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
```

In [24]:
```
def vif(preds):
    vif_data = pd.DataFrame()
    vif_data["Predictor"] = preds.columns
    vif_data["VIF"] = [variance_inflation_factor(preds.values, i) for i in range(len(preds.columns))]
    return vif_data
```

In [25]:
```
vif(X_new)
```

Out[25]:

	Predictor	VIF
0	const	1.000000
1	crim	1.789704
2	zn	2.239229
3	chas	1.059819
4	nox	3.778011
5	rm	1.834806
6	dis	3.443420
7	rad	6.861126
8	tax	7.272386
9	ptratio	1.757681
10	b	1.341559
11	lstat	2.581984

There are several predictors that is linked with others

In [26]:
```
X_new_2 = X_new.drop(columns=["tax"])
vif(X_new_2)
```

Out[26]:

	Predictor	VIF
0	const	1.000000
1	crim	1.787963
2	zn	2.154054
3	chas	1.052428
4	nox	3.564036
5	rm	1.806735
6	dis	3.410587
7	rad	2.776775
8	ptratio	1.717222
9	b	1.338982
10	lstat	2.579040

It looks much better
VIF of *rad* reduced significantly and it means thar *tax* and *rad* was linked

Checking new model without *tax*

In [27]:
```
model_new_2 = sm.OLS(medv, X_new_2)
results_new_2 = model_new_2.fit()

print(results_new_2.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.734
Model:                            OLS   Adj. R-squared:                  0.729
Method:                 Least Squares   F-statistic:                     136.7
Date:                Tue, 13 Dec 2022   Prob (F-statistic):          1.84e-135
Time:                        14:52:36   Log-Likelihood:                -1505.0
No. Observations:                 506   AIC:                             3032.
Df Residuals:                     495   BIC:                             3079.
Df Model:                          10
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         22.5328      0.213    105.828      0.000      22.114      22.951
crim          -0.9018      0.285     -3.164      0.002      -1.462      -0.342
zn             0.8544      0.313      2.731      0.007       0.240       1.469
chas           0.7538      0.219      3.448      0.001       0.324       1.183
nox           -2.3540      0.402     -5.850      0.000      -3.145      -1.563
rm             2.7944      0.286      9.754      0.000       2.232       3.357
dis           -3.0098      0.394     -7.647      0.000      -3.783      -2.236
rad            1.1212      0.355      3.157      0.002       0.423       1.819
ptratio       -2.1972      0.279     -7.867      0.000      -2.746      -1.648
b              0.8856      0.247      3.591      0.000       0.401       1.370
lstat         -3.7715      0.342    -11.019      0.000      -4.444      -3.099
==============================================================================
Omnibus:                      166.907   Durbin-Watson:                   1.090
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              684.418
Skew:                           1.441   Prob(JB):                    2.40e-149
Kurtosis:                       7.915   Cond. No.                         5.02
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

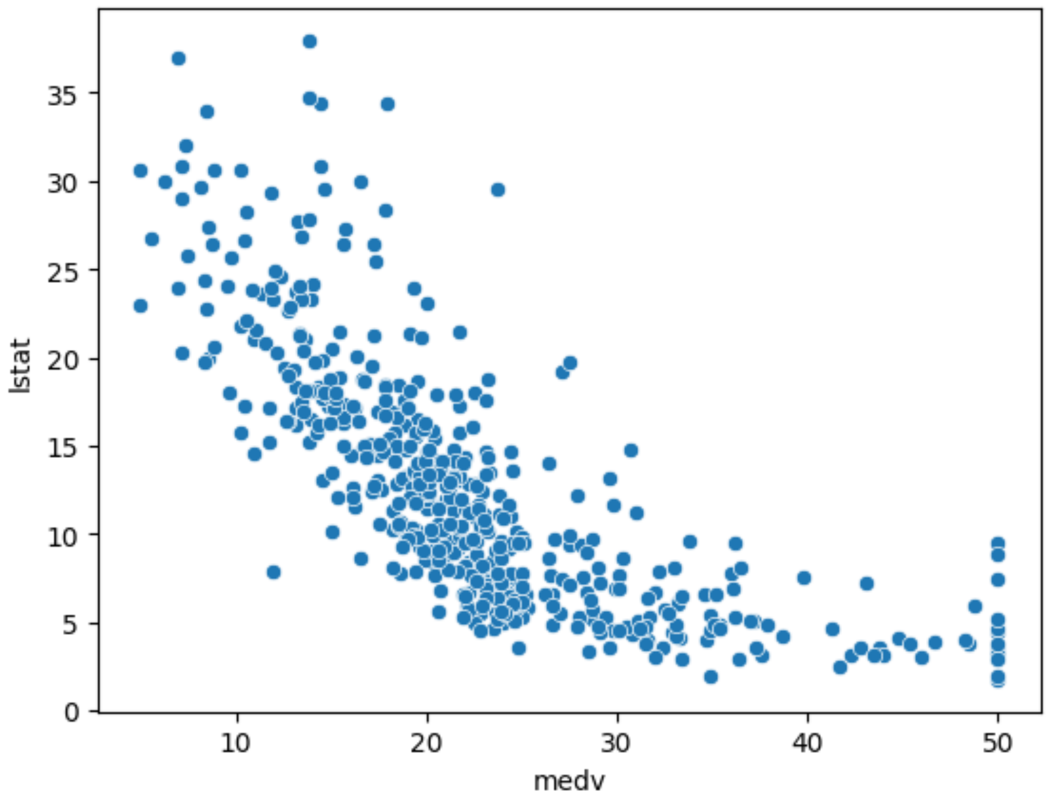
Statistics increased!

Now model looks pretty!

The largest modulo value of coefficient belongs to *lstat* predictor

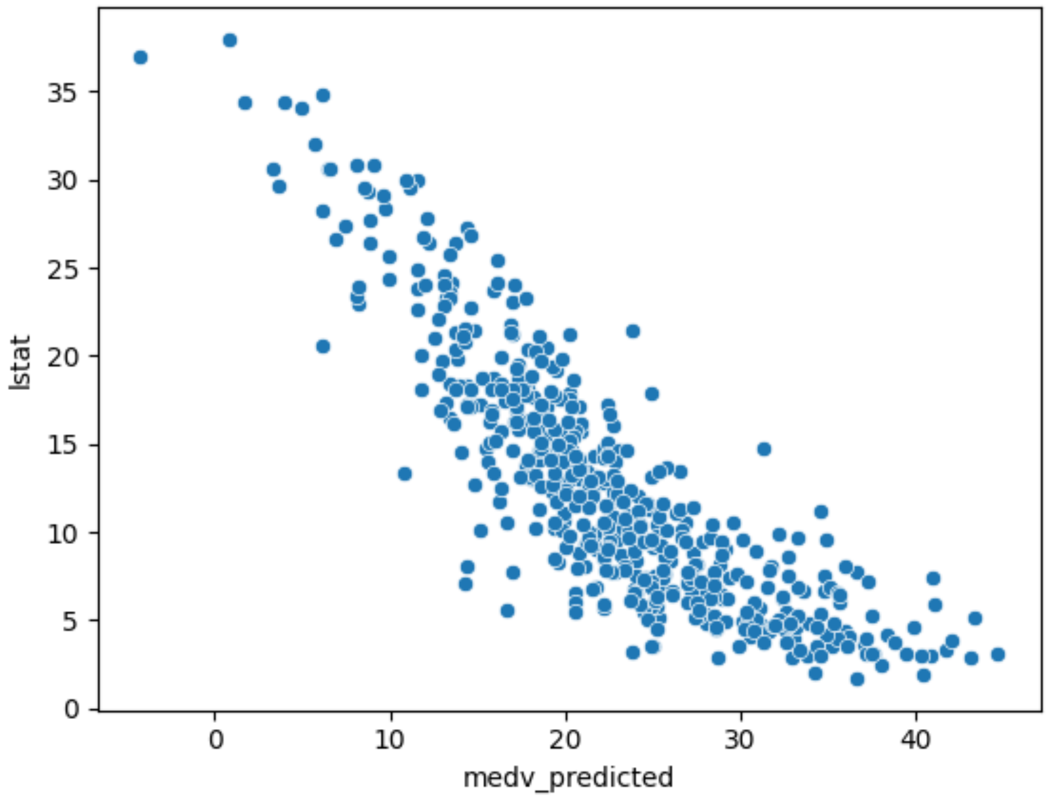
rm vs raw *medv*

In [28]:
```python
sns.scatterplot(x = medv, y = predictors.lstat);
```



lstat vs predicted *medv*

In [29]:
```python
sns.scatterplot(x = medv_predicted, y = predictors.lstat);
plt.xlabel('medv_predicted');
```



OPTIONAL

Let's see parameters of suburbs that have *medv* variable > mean + 1 sd - It will be named *data_medv_high*
And suburbs that have *medv* variable < mean + 1 sd - It will be named *data_medv_low*

In [30]:
```
mean_medv = medv.mean()
std_medv = medv.std()
```

In [31]:
```
treschold = mean_medv + std_medv
```

In [32]:
```
data_medv_high = data.loc[data.medv > treschold]
data_medv_low = data.loc[data.medv <= treschold]
```

Looking through correlation between medv and all predictors:

In [33]:
```
(pd.concat([medv, X_new_2.drop(columns=['const'])], axis=1)).corr().iloc[:,0]
```

Out[33]:
```
medv       1.000000
crim      -0.388305
zn         0.360445
chas       0.175260
nox       -0.427321
rm         0.695360
dis        0.249929
rad       -0.381626
ptratio   -0.507787
b          0.333461
lstat     -0.737663
Name: medv, dtype: float64
```

I will check predictors with high correclatioon values

One-sided Mann-Whitney U test for all predictors vs *medv*:

In [34]:
```
for col in data_medv_high.columns:
    print(col, '- in cheap suburbs greater')
    print(f"p-value - {mannwhitneyu(data_medv_low[col], data_medv_high[col], alternative='greater').pvalue}")
    print(col, '- in expensive suburbs greater')
    print(f"p-value - {mannwhitneyu(data_medv_low[col], data_medv_high[col], alternative='less').pvalue}")
    print()
```

```
crim - in cheap suburbs greater
p-value - 6.192291924409378e-07
crim - in expensive suburbs greater
p-value - 0.9999993835304415

zn - in cheap suburbs greater
p-value - 0.9999999999961751
zn - in expensive suburbs greater
p-value - 3.8556018089623235e-12

indus - in cheap suburbs greater
p-value - 1.062521626251081e-14
indus - in expensive suburbs greater
p-value - 0.9999999999999895

chas - in cheap suburbs greater
p-value - 0.9992560575911941
chas - in expensive suburbs greater
p-value - 0.000749130155310187

nox - in cheap suburbs greater
p-value - 3.640067266843077e-07
nox - in expensive suburbs greater
p-value - 0.9999996376482426

rm - in cheap suburbs greater
p-value - 1.0
rm - in expensive suburbs greater
p-value - 9.790460577535401e-33

age - in cheap suburbs greater
p-value - 0.00014914106723406078
age - in expensive suburbs greater
p-value - 0.9998513685389849

dis - in cheap suburbs greater
p-value - 0.9816522883378049
dis - in expensive suburbs greater
p-value - 0.01838761393683482

rad - in cheap suburbs greater
p-value - 0.0001496462931231421
rad - in expensive suburbs greater
p-value - 0.9998508754647684

tax - in cheap suburbs greater
p-value - 1.828698636468821e-10
tax - in expensive suburbs greater
p-value - 0.9999999998181772

ptratio - in cheap suburbs greater
p-value - 1.9715109522953002e-15
ptratio - in expensive suburbs greater
p-value - 0.999999999999998

b - in cheap suburbs greater
p-value - 0.9325626166258136
b - in expensive suburbs greater
p-value - 0.0675538449832815

lstat - in cheap suburbs greater
p-value - 1.1394403179327003e-31
lstat - in expensive suburbs greater
p-value - 1.0

medv - in cheap suburbs greater
p-value - 1.0
medv - in expensive suburbs greater
p-value - 5.358202079577759e-41
```

I'm very picky and suggest that predictors *b, rad, dis, age and chas* not significantly affects the cost of house

Testing *lstat*

Also, according to summary of last model, the most important parameter for higher cost is lower status of the population (percent)

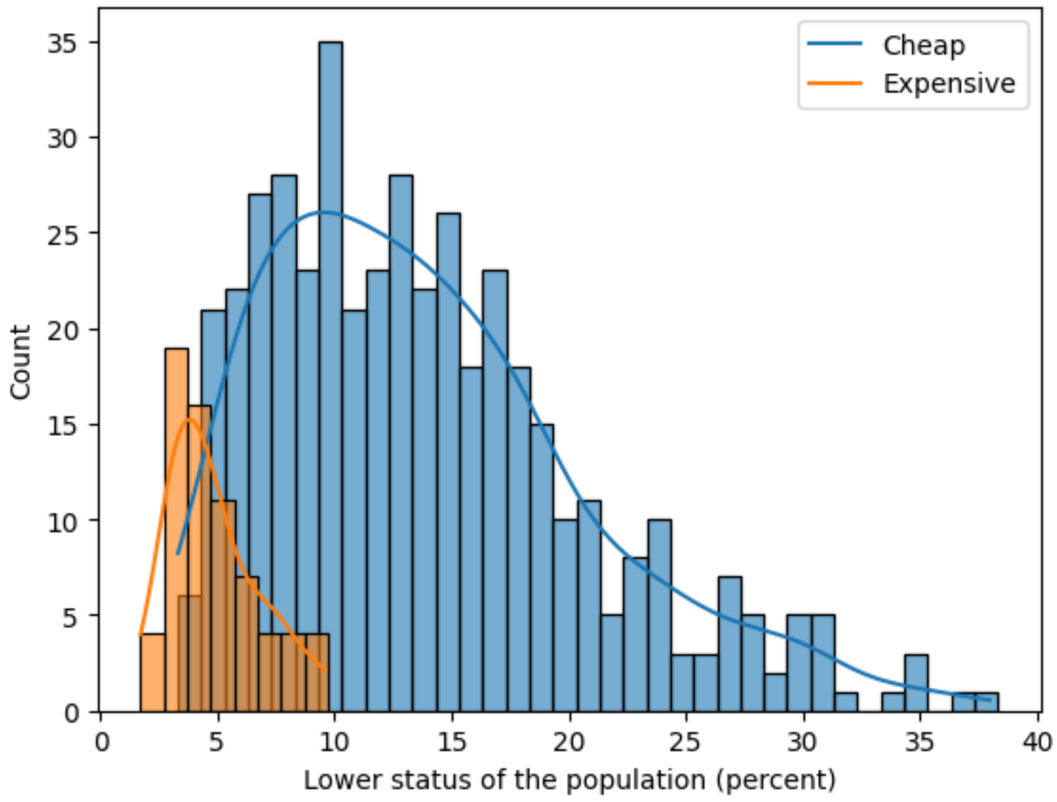
In [35]:
```
data_medv_low.lstat.describe()
```

Out[35]:
```
count    437.000000
mean      13.883547
std        6.880995
min        3.330000
25%        8.610000
50%       12.800000
75%       17.640000
max       37.970000
Name: lstat, dtype: float64
```

In [36]:
```
data_medv_high.lstat.describe()
```

Out[36]:
```
count    69.000000
mean      4.860000
std       1.941909
min       1.730000
25%       3.320000
50%       4.450000
75%       6.050000
max       9.590000
Name: lstat, dtype: float64
```

In [37]:
```
b1 = sns.histplot(data_medv_low.lstat, kde=True,
                  alpha=0.6, binwidth=1);
b2 = sns.histplot(data_medv_high.lstat, kde=True,
                  alpha=0.6, binwidth=1);
plt.xlabel('Lower status of the population (percent)');
plt.legend(['Cheap', 'Expensive']);
```



In [38]:
```
pval_mann_lstat = mannwhitneyu(data_medv_high.lstat, data_medv_low.lstat, alternative='less').pvalue
print('Expensive suburbs have greater value of lower status of the population (percent) then between cheap suburbs\n
      f'with pvalue of Mann-Whitney U test: {pval_mann_lstat}')
```

```
Expensive suburbs have greater value of lower status of the population (percent) then between cheap suburbs
with pvalue of Mann-Whitney U test: 1.1394403179327003e-31
```

Testing *rm*

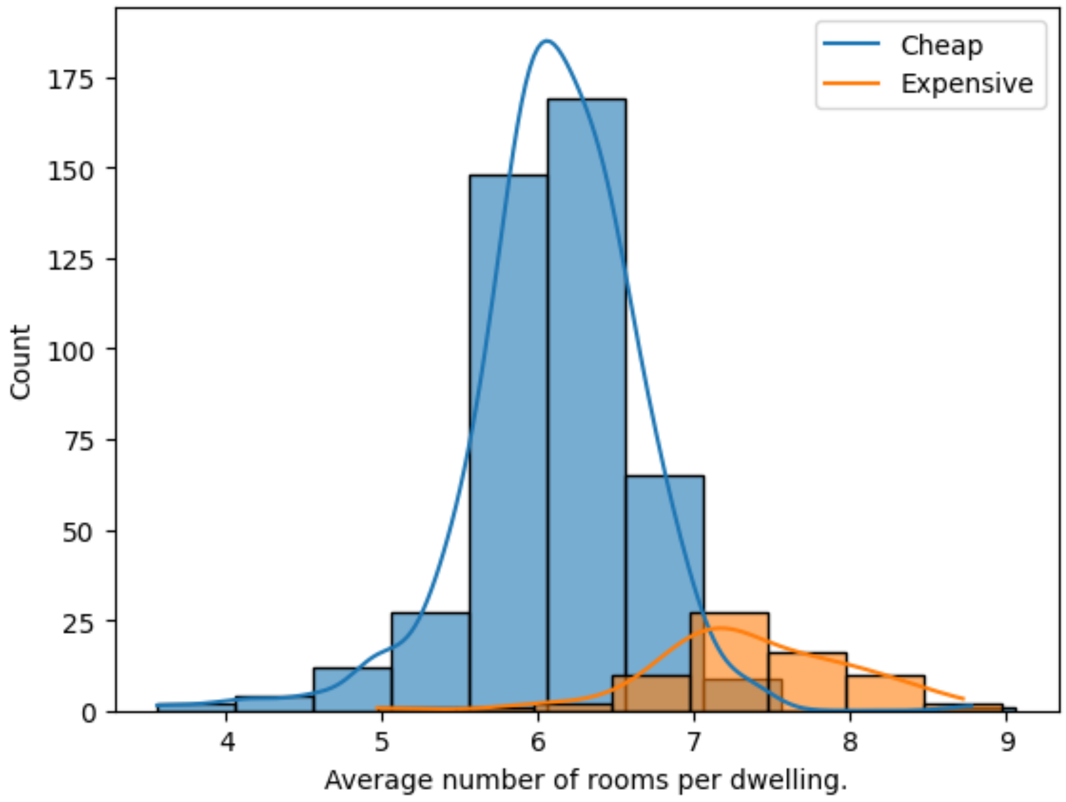
In [39]:
```
data_medv_low.rm.describe()
```

Out[39]:
```
count    437.000000
mean       6.112847
std        0.536631
min        3.561000
25%        5.857000
50%        6.127000
75%        6.431000
max        8.780000
Name: rm, dtype: float64
```

In [40]:
```
data_medv_high.rm.describe()
```

Out[40]:
```
count    69.000000
mean      7.372623
std       0.655013
min       4.970000
25%       6.998000
50%       7.267000
75%       7.820000
max       8.725000
Name: rm, dtype: float64
```

In [41]:
```
b1 = sns.histplot(data_medv_low.rm, kde=True,
                  alpha=0.6, binwidth=0.5);
b2 = sns.histplot(data_medv_high.rm, kde=True,
                  alpha=0.6, binwidth=0.5);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Average number of rooms per dwelling.');
```



In [42]:
```
pval_mann_rm = mannwhitneyu(data_medv_low.dis, data_medv_high.rm, alternative='less').pvalue
print('Expensive suburbs have greater average number of rooms per dwelling than cheap ones',
      f'with one sided Mann-Whitney U test p-value: {pval_mann_rm}')
```

```
Expensive suburbs have greater average number of rooms per dwelling than cheap ones with one sided Mann-Whitney U te
st p-value: 7.653663514039866e-30
```

Testing *ptratio*

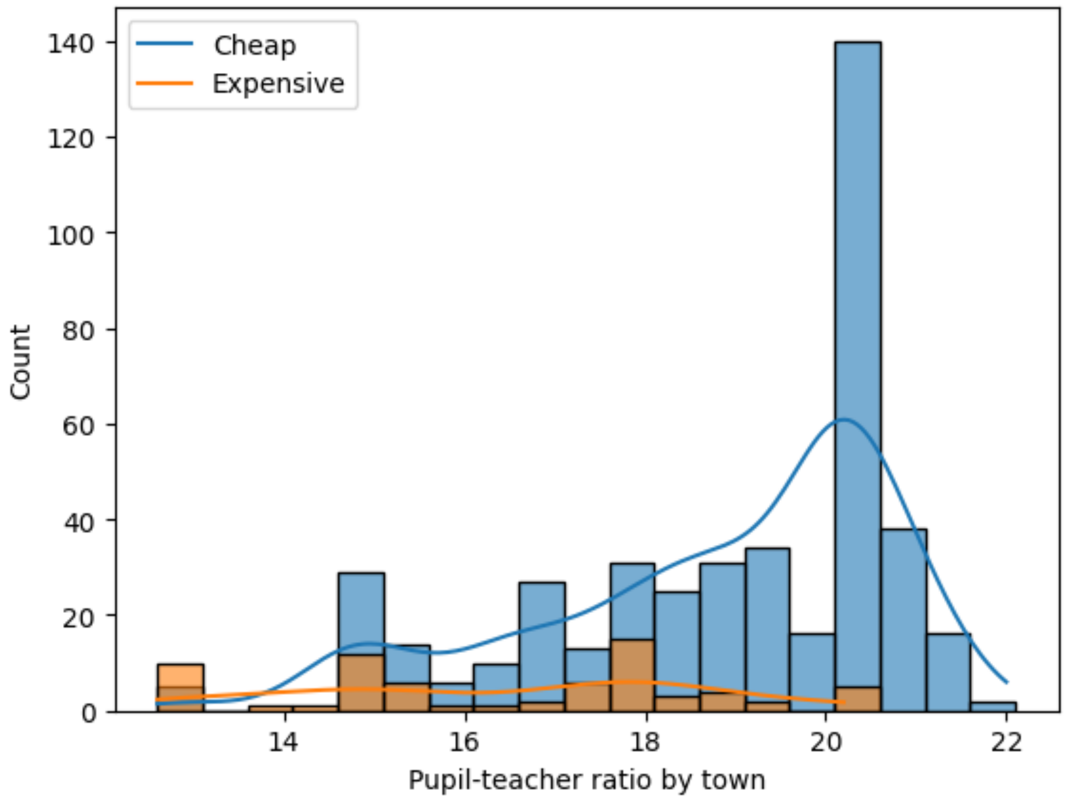
In [43]:
```
data_medv_low.ptratio.describe()
```

Out[43]:
```
count    437.000000
mean      18.778490
std        1.976883
min       12.600000
25%       17.800000
50%       19.200000
75%       20.200000
max       22.000000
Name: ptratio, dtype: float64
```

In [44]:
```
data_medv_high.ptratio.describe()
```

Out[44]:
```
count    69.000000
mean     16.410145
std       2.198806
min      12.600000
25%      14.700000
50%      17.400000
75%      17.900000
max      20.200000
Name: ptratio, dtype: float64
```

In [45]:
```
sns.histplot(data_medv_low.ptratio, kde=True, alpha=0.6, binwidth=0.5);
sns.histplot(data_medv_high.ptratio, kde=True, alpha=0.6, binwidth=0.5);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Pupil-teacher ratio by town');
# plt.ylim(0, 60);
```



In [46]:
```
pval_mann_ptratio = mannwhitneyu(data_medv_low.ptratio, data_medv_high.ptratio, alternative='greater').pvalue
print('Expensive suburbs have less Pupil-teacher ratio by town than cheap ones',
      f'with one sided Mann-Whitney U test p-value: {pval_mann_ptratio}')
```

```
Expensive suburbs have less Pupil-teacher ratio by town than cheap ones with one sided Mann-Whitney U test p-value:
1.9715109522953002e-15
```

After testing three most significant predictors I can say that higher cost of suburbs significantly depends on

1. lower status of population
2. number of rooms
3. pupil-teacher ratio

The first parameter is interesting because the population with "low status" simply cannot buy expensive houses. So the correlation is high.
The second parameter is clear - big house = big price.
The third is the dependence on education. The small number of students in one class is characteristic of private schools, which usually educate the children of wealthy parents.

Testing *nox*

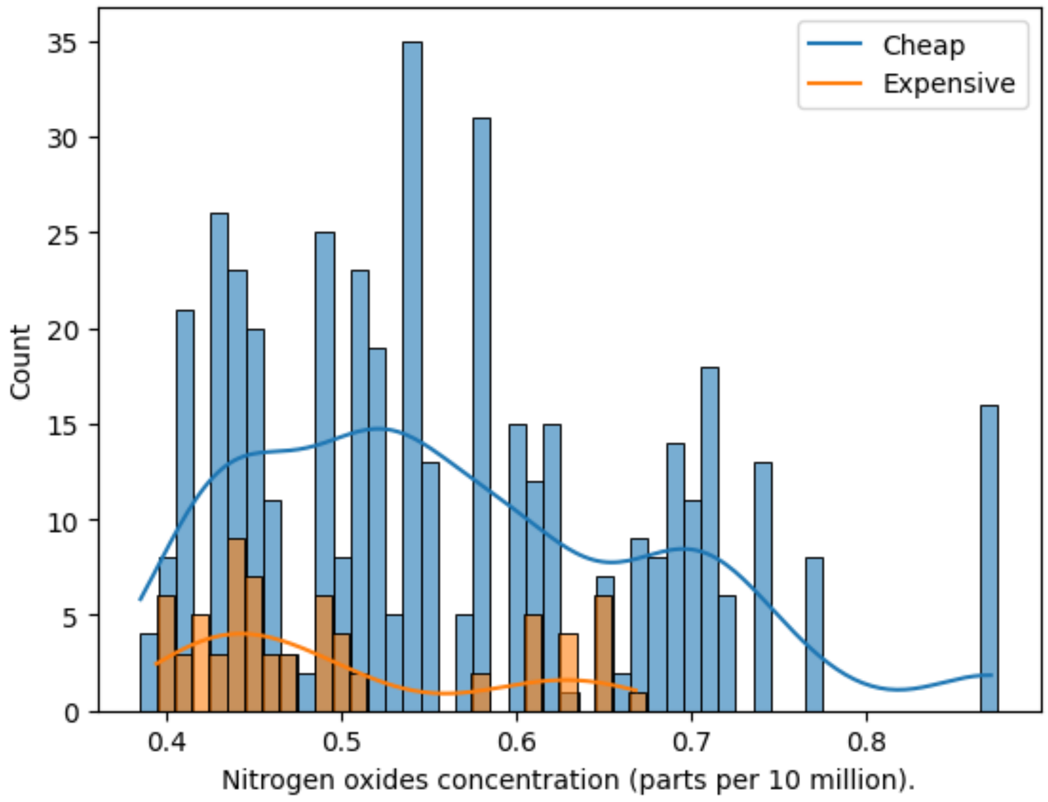
In [47]:
```
data_medv_low.nox.describe()
```

Out[47]:
```
count    437.000000
mean       0.564281
std        0.117315
min        0.385000
25%        0.464000
50%        0.538000
75%        0.647000
max        0.871000
Name: nox, dtype: float64
```

In [48]:
```
data_medv_high.nox.describe()
```

Out[48]:
```
count    69.000000
mean      0.493984
std       0.084527
min       0.394000
25%       0.437000
50%       0.458000
75%       0.575000
max       0.668000
Name: nox, dtype: float64
```

In [49]:
```
b1 = sns.histplot(data_medv_low.nox, kde=True,
                  alpha=0.6, binwidth=0.01);
b2 = sns.histplot(data_medv_high.nox, kde=True,
                  alpha=0.6, binwidth=0.01);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Nitrogen oxides concentration (parts per 10 million).');
# plt.ylim(0,15);
```



In [50]:
```
pval_mann_nox = mannwhitneyu(data_medv_low.nox, data_medv_high.nox, alternative='greater').pvalue
print('Expensive suburbs have less Nitrogen oxides concentration (parts per 10 million) than cheap ones',
      f'with one sided Mann-Whitney U test p-value: {pval_mann_nox}')
```

```
Expensive suburbs have less Nitrogen oxides concentration (parts per 10 million) than cheap ones with one sided Mann
-Whitney U test p-value: 3.640067266843077e-07
```

The same shape of distribution, but different mean value

Looks like it is two groups among all suburbs: with high nitrogen oxides concentration and lowe (two maximum peaks on the distribution)

Testing *crim*

In [51]:
```
data_medv_low.crim.describe()
```

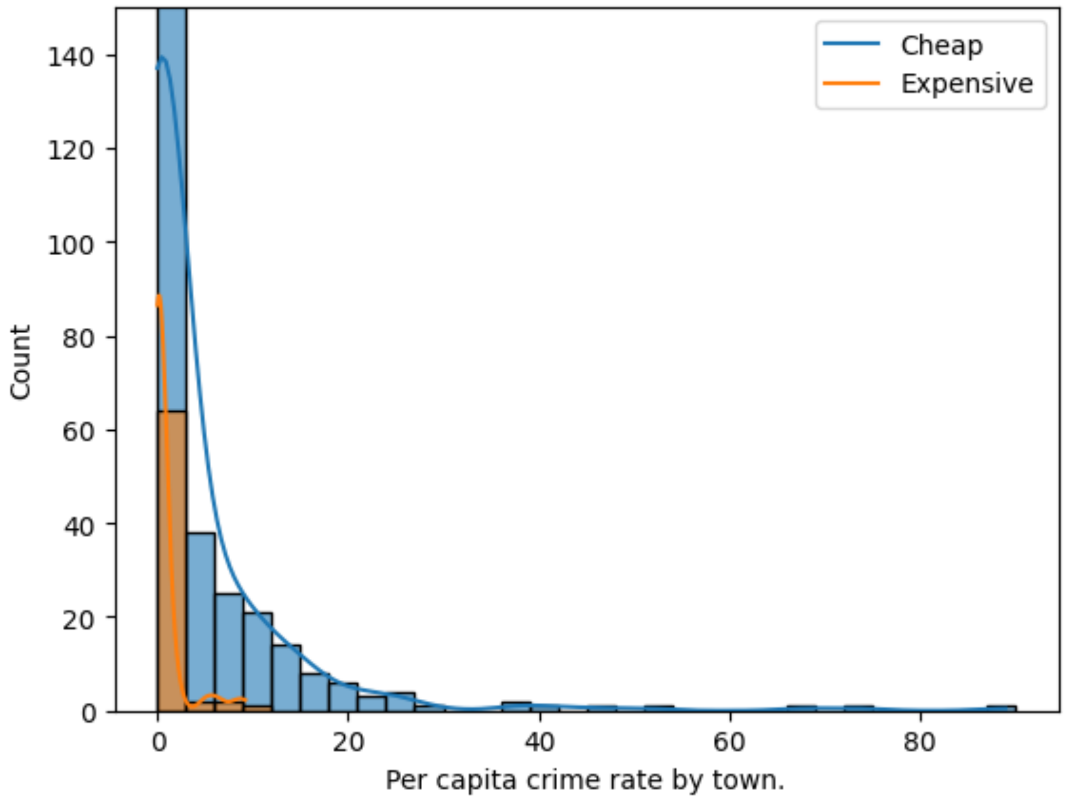
Out[51]:
```
count    437.000000
mean       4.063478
std        9.147717
min        0.006320
25%        0.092990
50%        0.290900
75%        4.541920
max       88.976200
Name: crim, dtype: float64
```

In [52]:
```
data_medv_high.crim.describe()
```

Out[52]:
```
count    69.000000
mean      0.763810
std       1.837559
min       0.009060
25%       0.037680
50%       0.086640
75%       0.534120
max       9.232300
Name: crim, dtype: float64
```

In [53]:
```
b1 = sns.histplot(data_medv_low.crim, kde=True,
                  alpha=0.6, binwidth=3);
b2 = sns.histplot(data_medv_high.crim, kde=True,
                  alpha=0.6, binwidth=3);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Per capita crime rate by town.');
plt.ylim(0, 150)
```

Out[53]:
```
(0.0, 150.0)
```



In [54]:
```
pval_mann_crim = mannwhitneyu(data_medv_low.crim, data_medv_high.crim, alternative='greater').pvalue
print('Expensive suburbs have less per capita crime rate by town than cheap ones',
      f'with one sided Mann-Whitney U test p-value: {pval_mann_crim}')
```

```
Expensive suburbs have less per capita crime rate by town than cheap ones with one sided Mann-Whitney U test p-value
: 6.192291924409378e-07
```

Cheap districs in some occausions have very high criminality rate (up to 89). Expensive ones have maximum 9 *crim*

Testing *zn*

In [55]:
```
data_medv_low.zn.describe()
```

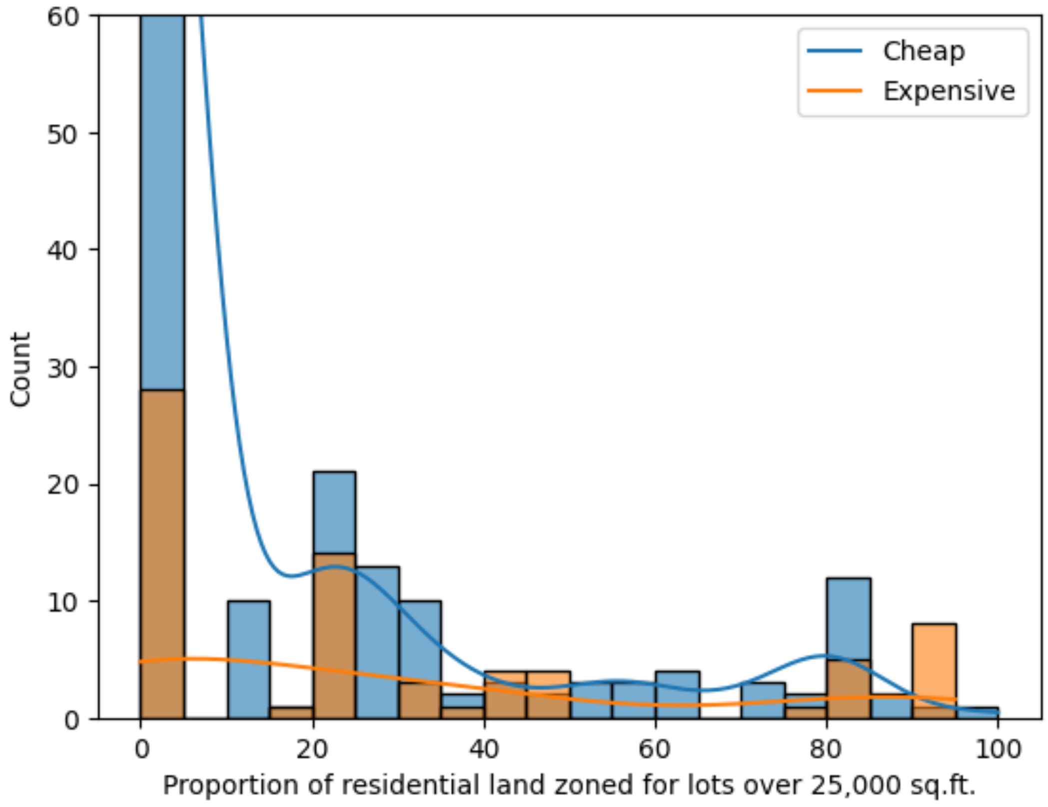
Out[55]:
```
count    437.000000
mean       8.599542
std       20.099033
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max      100.000000
Name: zn, dtype: float64
```

In [56]:
```
data_medv_high.zn.describe()
```

Out[56]:
```
count    69.000000
mean     28.869565
std      33.004529
min       0.000000
25%       0.000000
50%      20.000000
75%      45.000000
max      95.000000
Name: zn, dtype: float64
```

In [57]:
```
b1 = sns.histplot(data_medv_low.zn, kde=True,
                  alpha=0.6, binwidth=5);
b2 = sns.histplot(data_medv_high.zn, kde=True,
                  alpha=0.6, binwidth=5);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Proportion of residential land zoned for lots over 25,000 sq.ft.');
plt.ylim(0, 60)
```

Out[57]:
```
(0.0, 60.0)
```

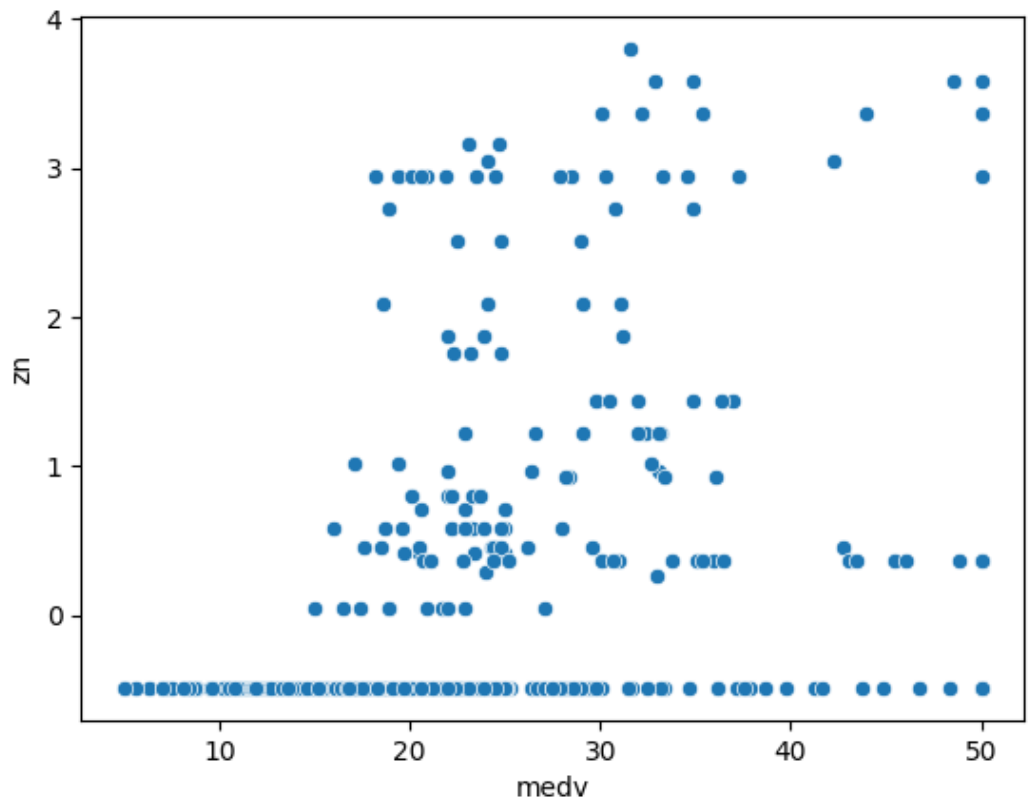


In [58]:
```
pval_mann_zn = mannwhitneyu(data_medv_low.zn, data_medv_high.zn, alternative='less').pvalue
print('Expensive suburbs have greater proportion of residential land zoned for lots over 25,000 sq.ft. than cheap on
      f'with one sided Mann-Whitney U test p-value: {pval_mann_zn}')
```

```
Expensive suburbs have greater proportion of residential land zoned for lots over 25,000 sq.ft. than cheap ones with
one sided Mann-Whitney U test p-value: 3.8556018089623235e-12
```

Scatter

In [59]:
```
sns.scatterplot(x=medv, y=scaled_preds.zn);
```



But there are a lot of zero values

Yes expensive houses have less zero *zn*, but it looks like there are simply less expensive houses.
However correlation exists: a lot of such zones => less houses in suburbs and more lands per house => higher cost

Testing indus

In [60]:
```
data_medv_low.indus.describe()
```

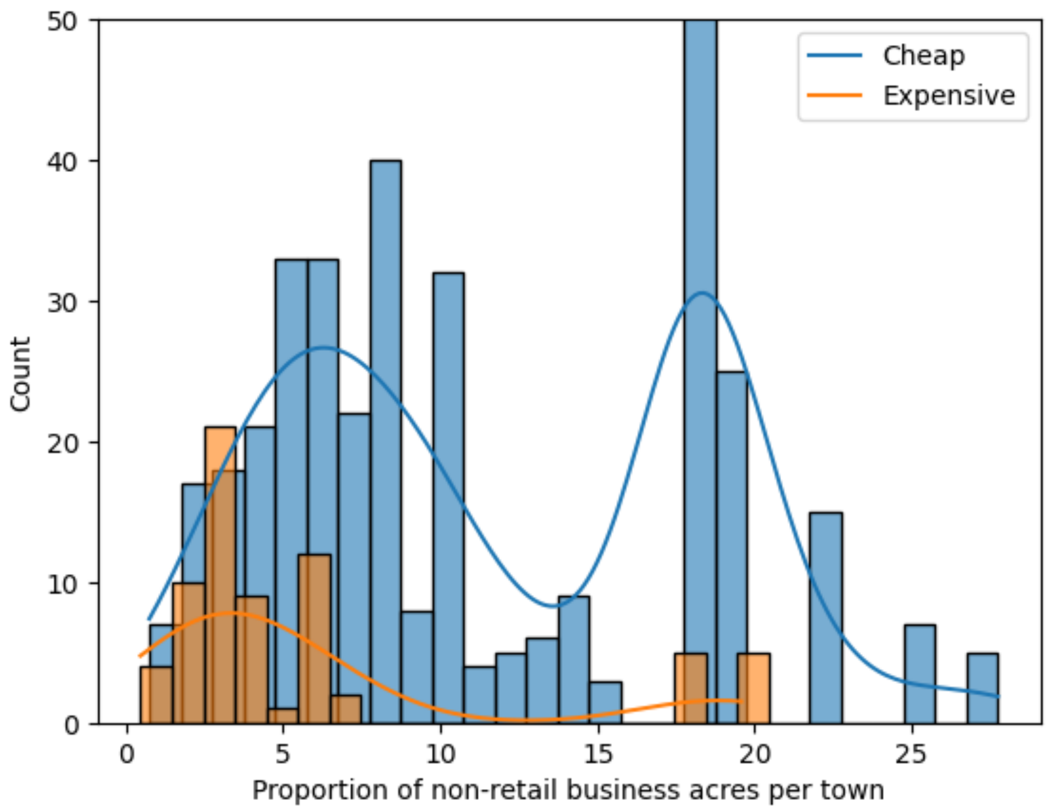
Out[60]:
```
count    437.000000
mean      11.975332
std        6.661414
min        0.740000
25%        6.060000
50%       10.010000
75%       18.100000
max       27.740000
Name: indus, dtype: float64
```

In [61]:
```
data_medv_high.indus.describe()
```

Out[61]:
```
count    69.000000
mean      5.825942
std       5.644957
min       0.460000
25%       2.460000
50%       3.440000
75%       6.200000
max      19.580000
Name: indus, dtype: float64
```

In [62]:
```
b1 = sns.histplot(data_medv_low.indus, kde=True,
                  alpha=0.6, binwidth=1);
b2 = sns.histplot(data_medv_high.indus, kde=True,
                  alpha=0.6, binwidth=1);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('Proportion of non-retail business acres per town');
plt.ylim(0, 50)
```

Out[62]: (0.0, 50.0)



In [63]:
```
pval_mann_indus = mannwhitneyu(data_medv_low.zn, data_medv_high.zn, alternative='less').pvalue
print('Expensive suburbs have greater proportion of non-retail business acres per town than cheap ones',
      f'with one sided Mann-Whitney U test p-value: {pval_mann_indus}')
```

```
Expensive suburbs have greater proportion of non-retail business acres per town than cheap ones with one sided Mann-Whitney U test p-value: 3.8556018089623235e-12
```

The plot looks like expensive houses should have a lower *indus* value, but the Mann-Whitney test shows us the opposite information. I think, that simply number of cheap houses is greater and it affected the results.

Also, it looks like there two grups of suburbs: with high and low values of non-retail business acres.

But some correlation exists!

Testing *tax*

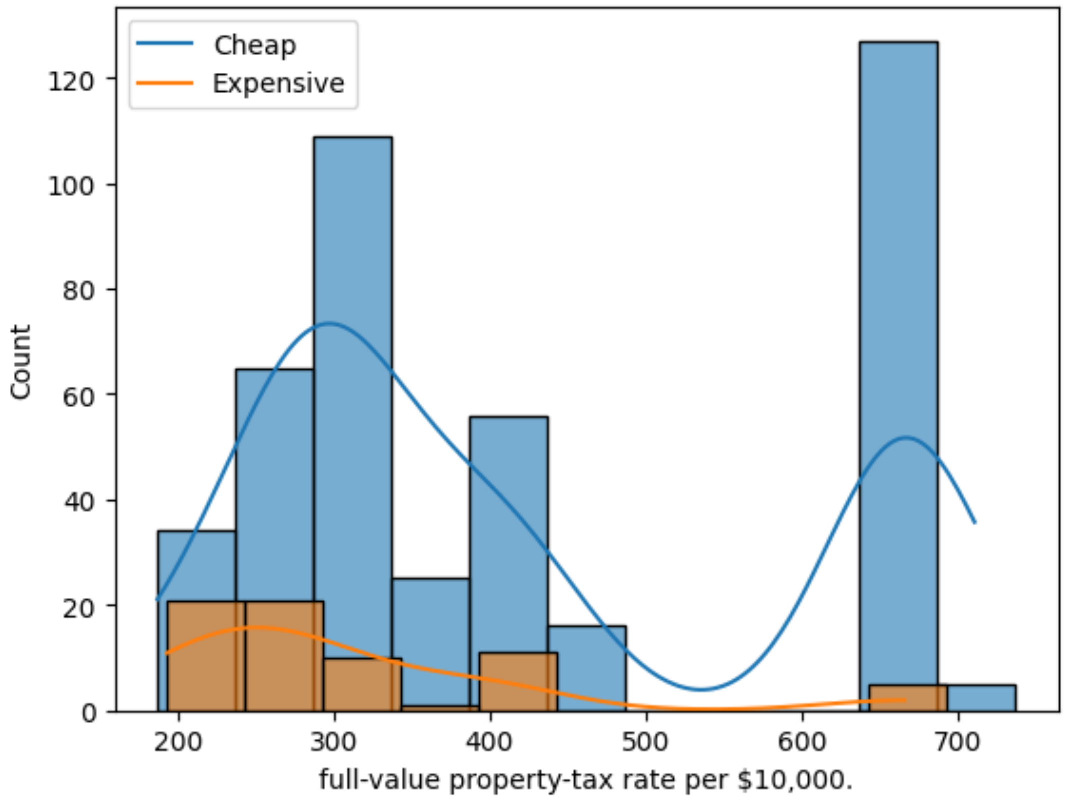
In [64]:
```
data_medv_low.tax.describe()
```

Out[64]:
```
count    437.000000
mean     424.109840
std      169.703456
min      187.000000
25%      289.000000
50%      358.000000
75%      666.000000
max      711.000000
Name: tax, dtype: float64
```

In [65]:
```
data_medv_high.tax.describe()
```

Out[65]:
```
count     69.000000
mean     307.710145
std      120.081725
min      193.000000
25%      224.000000
50%      264.000000
75%      330.000000
max      666.000000
Name: tax, dtype: float64
```

In [66]:
```
b1 = sns.histplot(data_medv_low.tax, kde=True,
                  alpha=0.6, binwidth=50);
b2 = sns.histplot(data_medv_high.tax, kde=True,
                  alpha=0.6, binwidth=50);
plt.legend(['Cheap', 'Expensive']);
plt.xlabel('full-value property-tax rate per \$10,000.');
```



The data don't look any different! + there are two peaks that indicate the existence of two groups depending on tax value

CONCLUSION

In my opinion, the most important aspects to consider when choosing an area to build a house are

- Lower status of the population

But dependency there may be inverse relationship - houses in suburbs are expensive and people with lower status cannot buy such houses

- Number of rooms

It is very significant parameter

- Pupil per teacher

Most of rich people want their children to study in private schools with high level of personal education

More comlicated parameters:

- Criminality

Criminality in rich subrubs is less on average, but cause of this distribution is because there are several "deviations" in cheap subrubs where criminality rates are extrimely high

- Nitrogen oxides concentrationLevel of nitrogen dioxide and non-retail business acres per town

I think this parameters depend on location of various factories and industrial compnies. Such districts are more ecologically friendly. There are no noises, smells and, in fact, people with lower status.

- Proportion of residential land zoned for lots over 25,000 sq.ft

Expensive subrubs have more lands per house, but it looks like there is small correlation. May be there some cheap subrubs that have small population with empty zones wuthout houses.

- Full-value property-tax rate per $10,000.

This parameter measures cost of public services. But correlation is not very significant. May be it is bacuse some cheap subrubs have big taxes and people live in cheap houses.

Best House ever:

1. With big nuber of rooms
2. With good educational insitutions nearby
3. No industrial complexes nearby
4. Big teritory
5. Good public services in town
6. No criminality in suburb