uczenie-nadzorowane-predykcja

May 12, 2025

0.0.1 Pobranie danych dla Boston Housing Dataset

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[4]:
```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/
 ↪housing.data"
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
         'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
dataset = pd.read_csv(url, sep='\s+', names=names)
```

0.0.2 Exploratory Data Analysis (EDA)

[5]:
```
print("Informacje o danych:")
print(dataset.info())
print("\nStatystyki opisowe:")
print(dataset.describe())

print("\nLiczba braków w danych:")
print(dataset.isnull().sum())
```

```
Informacje o danych:
<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    float64
 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(12), int64(2)
memory usage: 55.5 KB
None

Statystyki opisowe:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	\
count	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	
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	

	AGE	DIS	RAD	TAX	PTRATIO	B	\
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	
mean	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	
std	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	
min	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	
25%	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	
50%	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	
75%	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	
max	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	

	LSTAT	MEDV
count	506.000000	506.000000
mean	12.653063	22.532806
std	7.141062	9.197104
min	1.730000	5.000000
25%	6.950000	17.025000
50%	11.360000	21.200000
75%	16.955000	25.000000
max	37.970000	50.000000

```
Liczba braków w danych:
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
```

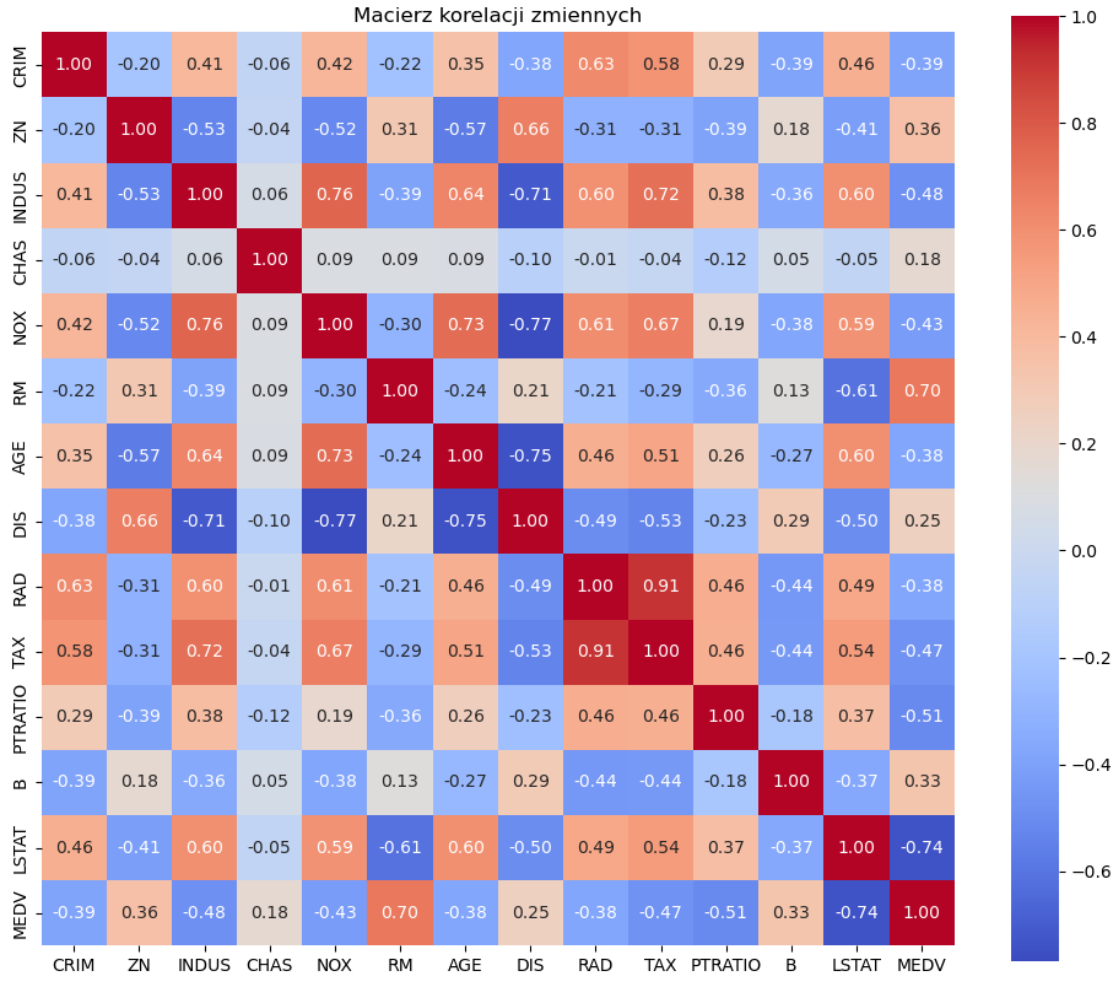
Struktura zbioru: - Zbiór danych zawiera 506 obserwacji i 14 kolumn. - Brak wartości brakujących – dane są kompletne, co upraszcza dalsze analizy.

Opis kolumn – Boston Housing Dataset

- **CRIM** – wskaźnik przestępczości na mieszkańca w danej dzielnicy
- **ZN** – procent terenów mieszkaniowych przeznaczonych pod działki większe niż 25 000 stóp kwadratowych
- **INDUS** – procent powierzchni miasta przeznaczonej na działalność pozahandlową (np. przemysł)
- **CHAS** – zmienna binarna: 1 jeśli działka graniczy z rzeką Charles, 0 w przeciwnym razie
- **NOX** – stężenie tlenków azotu (cząstek na 10 milionów)
- **RM** – średnia liczba pokoi na mieszkanie
- **AGE** – procent mieszkań zajmowanych przez właścicieli, które zostały zbudowane przed 1940 rokiem
- **DIS** – ważona odległość do pięciu głównych centrów zatrudnienia w Bostonie
- **RAD** – indeks dostępności do głównych autostrad
- **TAX** – pełna wartość podatku od nieruchomości na $10,000 wartości
- **PTRATIO** – stosunek liczby uczniów do liczby nauczycieli w danej dzielnicy
- **B** – $1000(Bk - 0.63)^2$, gdzie Bk to odsetek osób czarnoskórych w danym mieście (cecha przestarzała i kontrowersyjna)
- **LSTAT** – procent populacji o niskim statusie społecznym
- **MEDV** – mediana wartości domów zajmowanych przez właścicieli (w tysiącach dolarów)

[6]:
```
corr_matrix = dataset.corr()

plt.figure(figsize=(12, 10))
```

```
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", square=True)
plt.title("Macierz korelacji zmiennych")
plt.show()
```



[7]:
```
high_corr = corr_matrix.abs() > 0.7
print("\nWysoka współliniowość (|r| > 0.7):")
print(corr_matrix[high_corr & (corr_matrix != 1.0)])
```

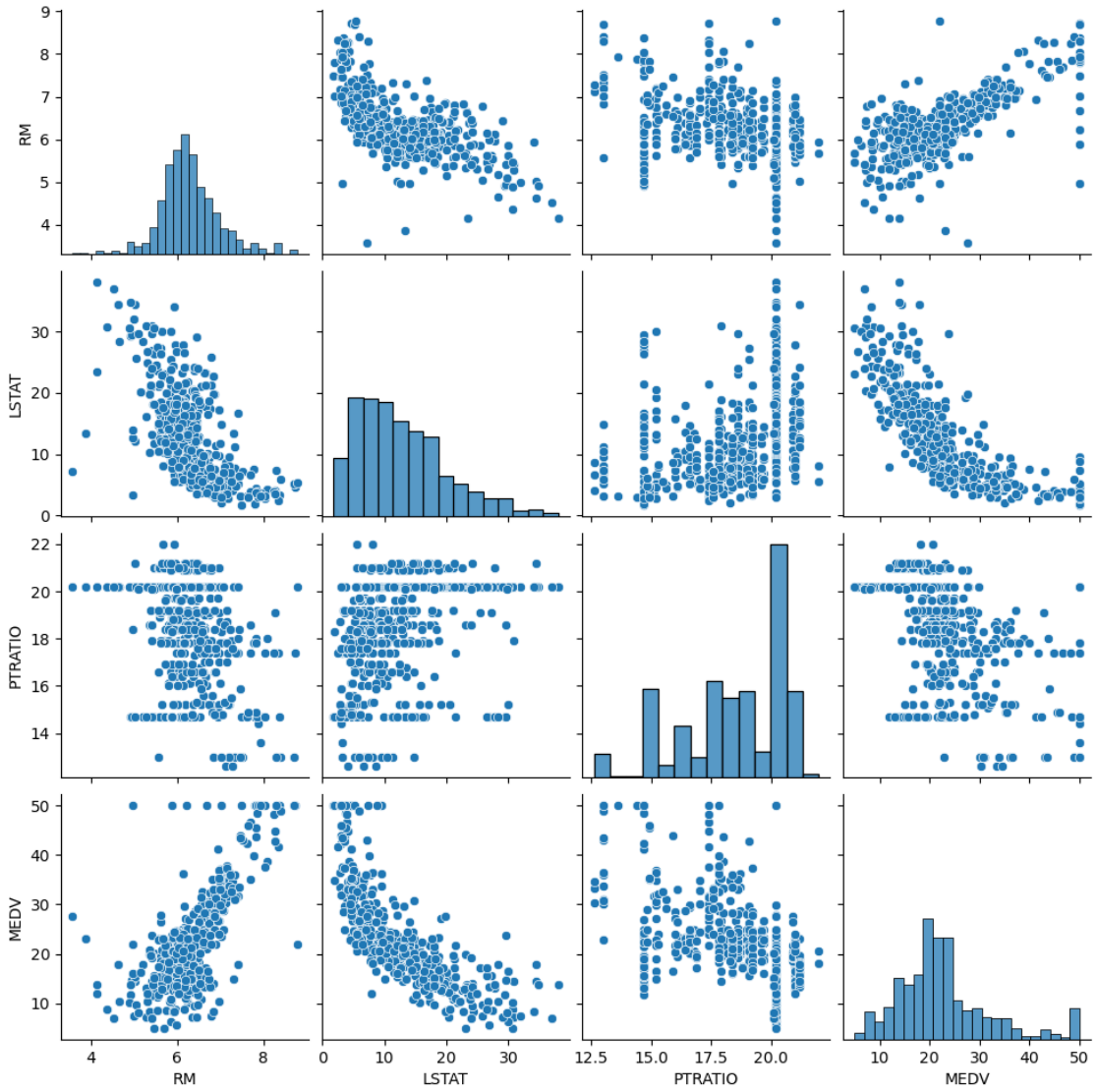
```
Wysoka współliniowość (|r| > 0.7):
         CRIM  ZN     INDUS  CHAS       NOX  RM       AGE       DIS       RAD  \
CRIM      NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
ZN        NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
INDUS     NaN NaN       NaN   NaN  0.763651 NaN       NaN -0.708027       NaN
CHAS      NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
NOX       NaN NaN  0.763651   NaN       NaN NaN  0.731470 -0.769230       NaN
RM        NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
```

```
AGE       NaN NaN       NaN   NaN  0.731470 NaN       NaN -0.747881       NaN
DIS       NaN NaN -0.708027   NaN -0.769230 NaN -0.747881       NaN       NaN
RAD       NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
TAX       NaN NaN  0.720760   NaN       NaN NaN       NaN       NaN  0.910228
PTRATIO   NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
B         NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
LSTAT     NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN
MEDV      NaN NaN       NaN   NaN       NaN NaN       NaN       NaN       NaN

               TAX  PTRATIO   B     LSTAT      MEDV
CRIM           NaN      NaN NaN       NaN       NaN
ZN             NaN      NaN NaN       NaN       NaN
INDUS     0.720760      NaN NaN       NaN       NaN
CHAS           NaN      NaN NaN       NaN       NaN
NOX            NaN      NaN NaN       NaN       NaN
RM             NaN      NaN NaN       NaN       NaN
AGE            NaN      NaN NaN       NaN       NaN
DIS            NaN      NaN NaN       NaN       NaN
RAD       0.910228      NaN NaN       NaN       NaN
TAX            NaN      NaN NaN       NaN       NaN
PTRATIO        NaN      NaN NaN       NaN       NaN
B              NaN      NaN NaN       NaN       NaN
LSTAT          NaN      NaN NaN       NaN -0.737663
MEDV           NaN      NaN NaN -0.737663       NaN
```

W analizie współczynników korelacji wykryto kilka par cech, które wykazują **wysoką współliniowość** ($|r| > 0.7$). Poniżej przedstawiono istotne zależności wraz z wyjaśnieniem znaczenia zmiennych:

- `RAD` (indeks dostępności do autostrad) i `TAX` (stawka podatku od nieruchomości) – $\mathbf{r = 0.91}$
  - Wskazuje na silną zależność między bliskością dróg a wysokością podatków.
- `NOX` (stężenie tlenków azotu) i `INDUS` (udział terenów przemysłowych) – $\mathbf{r = 0.76}$
  - Obszary przemysłowe mają wyższe zanieczyszczenie powietrza.
- `DIS` (odległość od centrów zatrudnienia) i `NOX` – $\mathbf{r = -0.77}$
  - Im dalej od centrum zatrudnienia, tym czystsze powietrze.
- `DIS` i `AGE` (procent starszych domów) – $\mathbf{r = -0.75}$
  - Starsze domy znajdują się bliżej od centrów zatrudnienia .
- `INDUS` i `TAX` – $\mathbf{r = 0.72}$
  - Obszary przemysłowe mają wyższe podatki od nieruchomości.
- `LSTAT` (procent populacji o niższym statusie społecznym) i `MEDV` (mediana wartości domów) – $\mathbf{r = -0.74}$
  - W dzielnicach o niższym statusie społecznym ceny domów są niższe.

[8]:
```
sns.pairplot(dataset[['RM', 'LSTAT', 'PTRATIO', 'MEDV']])
plt.show()
```

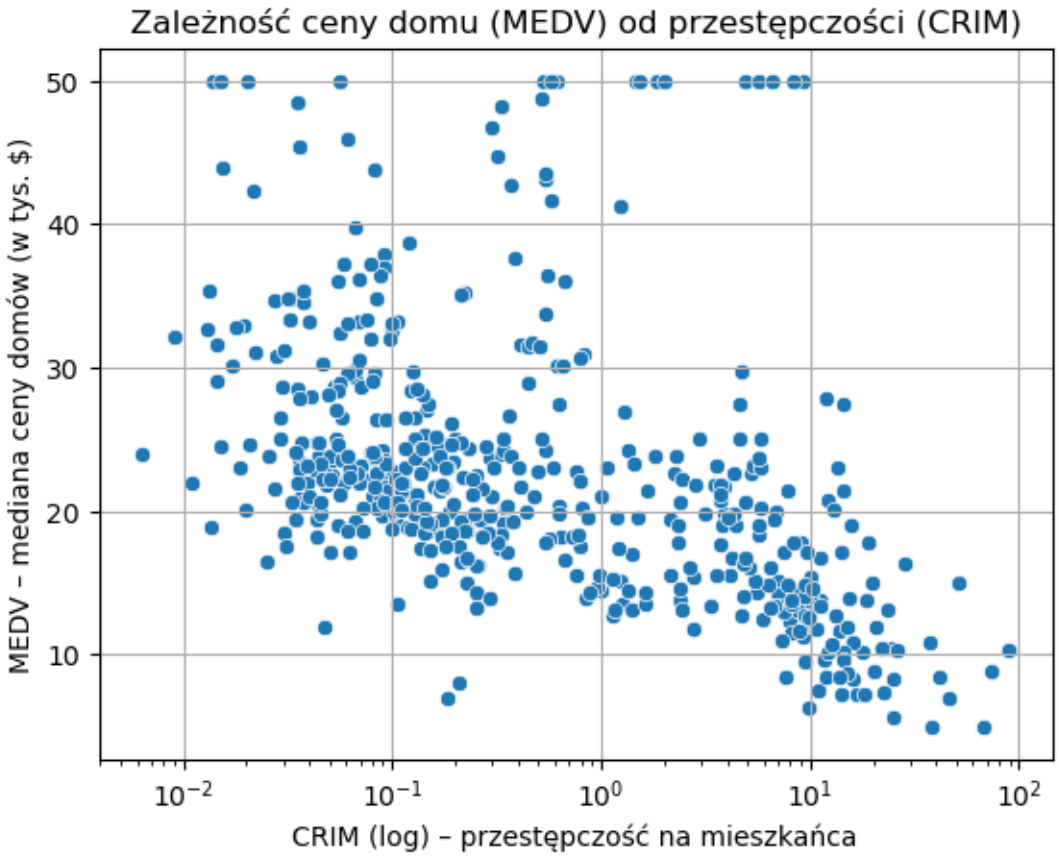


Na podstawie macierzy wykresów par (pairplot) dla zmiennych `RM`, `LSTAT`, `PTRATIO` i `MEDV` można zauważyć:

- RM (liczba pokoi na mieszkanie) ma dodatnią korelację z MEDV (ceną domu): im więcej pokoi, tym wyższa cena nieruchomości.
- LSTAT (procent osób o niższym statusie społecznym) ma silnie ujemną korelację z MEDV: w rejonach o niższym statusie społecznym ceny domów są niższe.

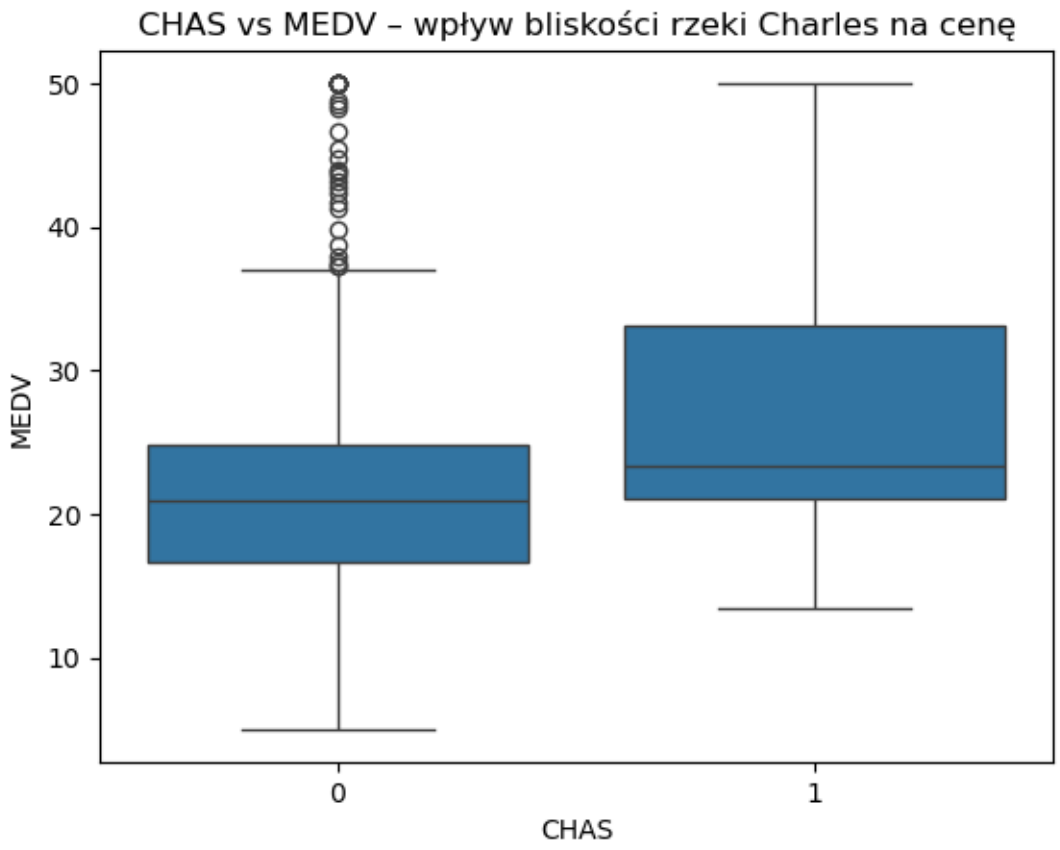
[15]:
```
sns.scatterplot(x='CRIM', y='MEDV', data=dataset)
plt.xscale('log')
plt.title('Zależność ceny domu (MEDV) od przestępczości (CRIM)')
plt.xlabel('CRIM (log) - przestępczość na mieszkańca')
plt.ylabel('MEDV - mediana ceny domów (w tys. $)')
plt.grid(True)
```

```
plt.show()
```



- Po zastosowaniu skali logarytmicznej dla `CRIM` (wskaźnik przestępczości), zależność między `CRIM` a `MEDV` (ceną domu) staje się **bardziej czytelna i uporządkowana**.
- Wyraźnie widać, że **przy niskich wartościach przestępczości** ($\log(\text{CRIM}) < 0$) dominuje większe zróżnicowanie cen — w tym **najwyższe ceny nieruchomości**.
- Wraz ze wzrostem `CRIM` (czyli przestępczości), mediana cen (`MEDV`) **systematycznie maleje**.

[11]:
```
sns.boxplot(x='CHAS', y='MEDV', data=dataset)
plt.title("CHAS vs MEDV - wpływ bliskości rzeki Charles na cenę")
plt.show()
```



- Zmienna `CHAS` (0 = brak bliskości rzeki, 1 = sąsiedztwo rzeki Charles) ma wpływ na `MEDV` (cenę domów).
- Mediana cen domów w pobliżu rzeki (`CHAS = 1`) jest wyraźnie wyższa niż poza jej sąsiedztwem.
- Wskazuje to, że **lokalizacja przy rzece może zwiększać wartość nieruchomości**.

0.0.3 Podział danych na zbiór treningowy i testowy.

[9]:
```
from sklearn.model_selection import train_test_split

X = dataset.drop('MEDV', axis=1)
y = dataset['MEDV']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

0.0.4 Zbadanie różnych modeli takich jak regresja liniowa i XGBoost.

[10]:
```
# Regresja liniowa
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


lr = LinearRegression()
lr.fit(X_train, y_train)

# XGBoost
import xgboost as xgb

xg_reg = xgb.XGBRegressor(
    objective='reg:squarederror',
    colsample_bytree=0.3,
    learning_rate=0.1,
    max_depth=5,
    alpha=10,
    n_estimators=10
)
xg_reg.fit(X_train, y_train)

y_pred_xg = xg_reg.predict(X_test)
print('XGBoost - MSE: ', mean_squared_error(y_test, y_pred_xg))
print('XGBoost - MAE: ', mean_absolute_error(y_test, y_pred_xg))
print('XGBoost - R2: ', r2_score(y_test, y_pred_xg))
```

```
XGBoost - MSE:  34.38827278702372
XGBoost - MAE:  3.8391113281249996
XGBoost - R2:  0.531071883856806
```

0.0.5 Zastosowanie GridSearchCV, aby dobrać optymalne hiperparametry dla modelu XGBoost. Można użyć różnych metryk ewaluacyjnych, takich jak MSE, RMSE, MAE, R2.

[12]:
```
# GridSearchCV dla XGBoost
from sklearn.model_selection import GridSearchCV

params = {
    'learning_rate': [0.01, 0.1, 0.3,0.15],
    'max_depth': [2,3, 5, 7,10],
    'n_estimators': [50, 100, 200,150,300]
}

xg_reg = xgb.XGBRegressor(objective='reg:squarederror', colsample_bytree=0.3)
```

```
grid = GridSearchCV(estimator=xg_reg, param_grid=params, cv=5,␣
 ↪scoring='neg_mean_squared_error') #wybiera najlepszą kombinacje parametrów
grid.fit(X_train, y_train)

print("Best score: %f using params: %s" % (grid.best_score_, grid.best_params_))
```

```
Best score: -13.113346 using params: {'learning_rate': 0.1, 'max_depth': 2,
'n_estimators': 300}
```

Porównanie wyników różnych modeli na zbiorze testowym, wykorzystując wybrane metryki ewaluacyjne.

[13]:
```
# Regresja liniowa
y_pred_lr = lr.predict(X_test)
print('Linear Regression - MSE: ', mean_squared_error(y_test, y_pred_lr))
print('Linear Regression - RMSE: ',  np.sqrt(mean_absolute_error(y_test,␣
 ↪y_pred_lr)))
print('Linear Regression - MAE: ', mean_absolute_error(y_test, y_pred_lr))
print('Linear Regression - R2: ', r2_score(y_test, y_pred_lr))

# XGBoost
y_pred_xg = grid.predict(X_test) #używa najlepszego zestawu parametrów ktory␣
 ↪wybral poprzednio
print('XGBoost - MSE: ', mean_squared_error(y_test, y_pred_xg))
print('XGBoost - RMSE: ', np.sqrt(mean_squared_error(y_test, y_pred_xg)))
print('XGBoost - MAE: ', mean_absolute_error(y_test, y_pred_xg))
print('XGBoost - R2: ', r2_score(y_test, y_pred_xg))
```

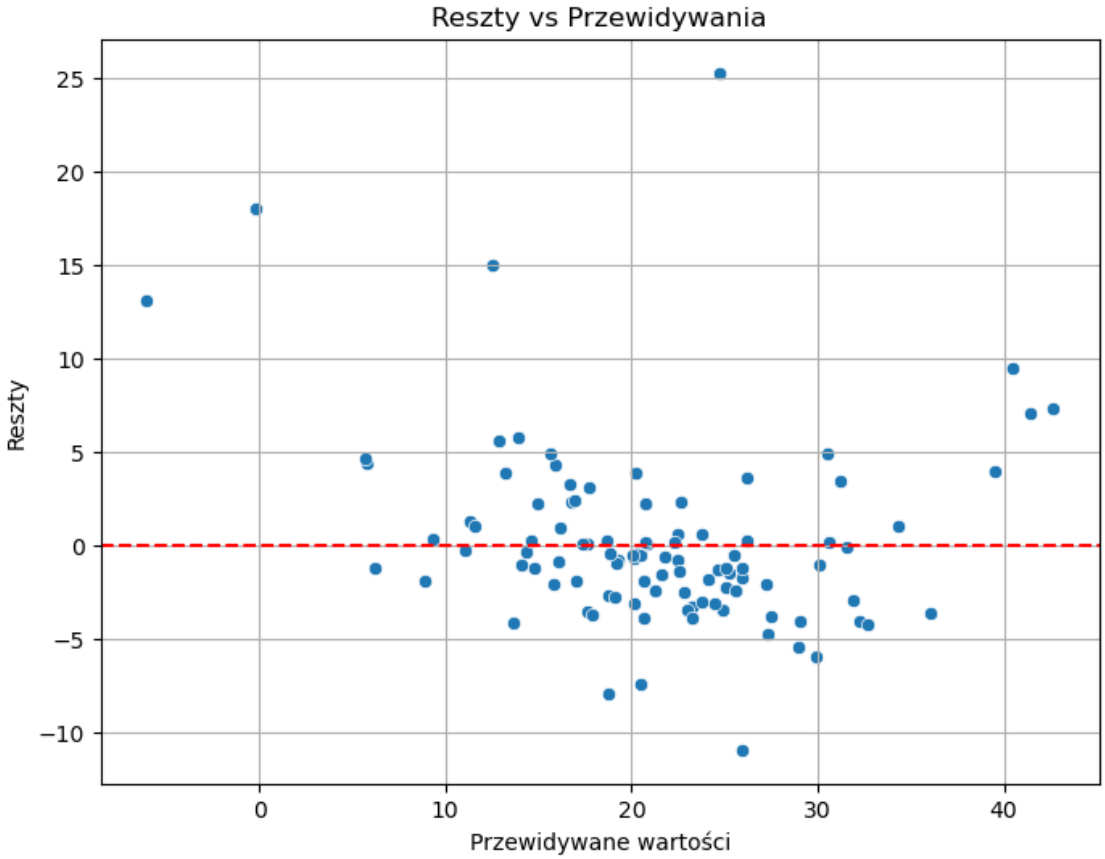
```
Linear Regression - MSE:  24.291119474973478
Linear Regression - RMSE:  1.7858028911074821
Linear Regression - MAE:  3.1890919658878416
Linear Regression - R2:  0.6687594935356326
XGBoost - MSE:  11.347675488068111
XGBoost - RMSE:  3.368631100026851
XGBoost - MAE:  2.136557480868171
XGBoost - R2:  0.8452599197935865
```

Wnioski

- Wykorzystano GridSearchCV do doboru hiperparametrów dla modelu XGBoost.
- Najlepsze parametry to: `learning_rate = 0.1`, `max_depth = 2`, `n_estimators = 300`
- Najlepszy wynik walidacji krzyżowej (CV): **MSE = 13.11**
- Wyniki na zbiorze testowym:
  - **Regresja liniowa**: MSE = 24.29, MAE = 3.19, R^2 = 0.669, RMSE = 4.93
  - **XGBoost**: MSE = 11.35, MAE = 2.14, R^2 = 0.845, RMSE = 3.37
- **Wniosek**: Model XGBoost zautomatyzowany przy pomocy GridSearchCV znacząco przewyższa regresję liniową pod względem dokładności predykcji.

0.0.6 Sprawdzenie, czy spełnione są założenia dla regresji liniowej

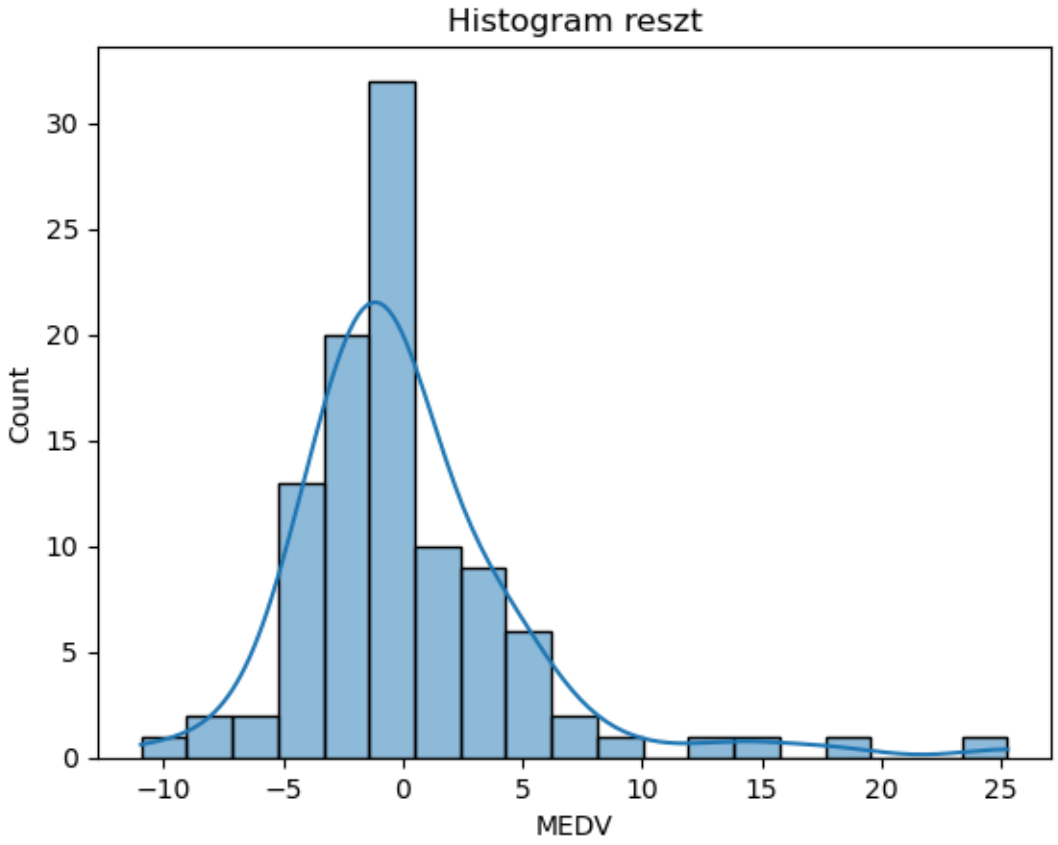
[14]:
```
residuals = y_test - y_pred_lr

plt.figure(figsize=(8, 6))
sns.scatterplot(x=y_pred_lr, y=residuals)
plt.axhline(0, color='red', linestyle='--')
plt.xlabel('Przewidywane wartości')
plt.ylabel('Reszty')
plt.title('Reszty vs Przewidywania')
plt.grid(True)
plt.show()
```



Sprawdzenie założeń regresji – reszty

- Na wykresie nie widać wyraźnego wzoru, co sugeruje, że założenie liniowości jest raczej spełnione.
- Rozrzut reszt jest dość równomierny wokół poziomu 0 (czerwona linia), ale:
  - Widać kilka punktów odstających (outliers), zwłaszcza dla niskich i wysokich wartości przewidywanych.

[15]:
```
import scipy.stats as stats

sns.histplot(residuals, kde=True)
plt.title('Histogram reszt')
plt.show()
```



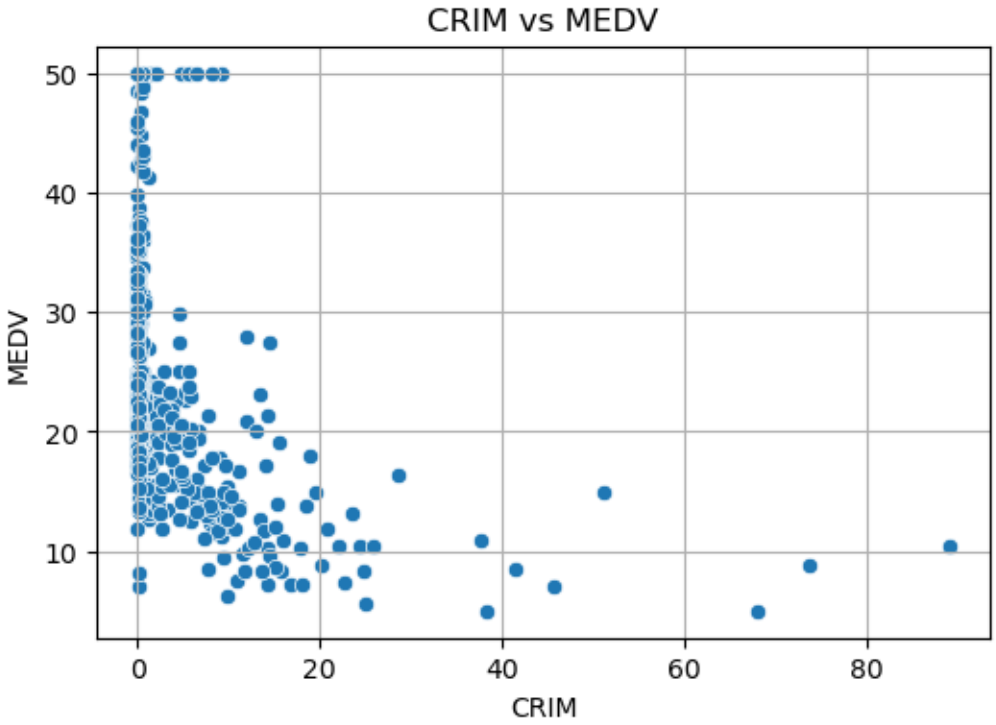
Sprawdzenie normalności reszt

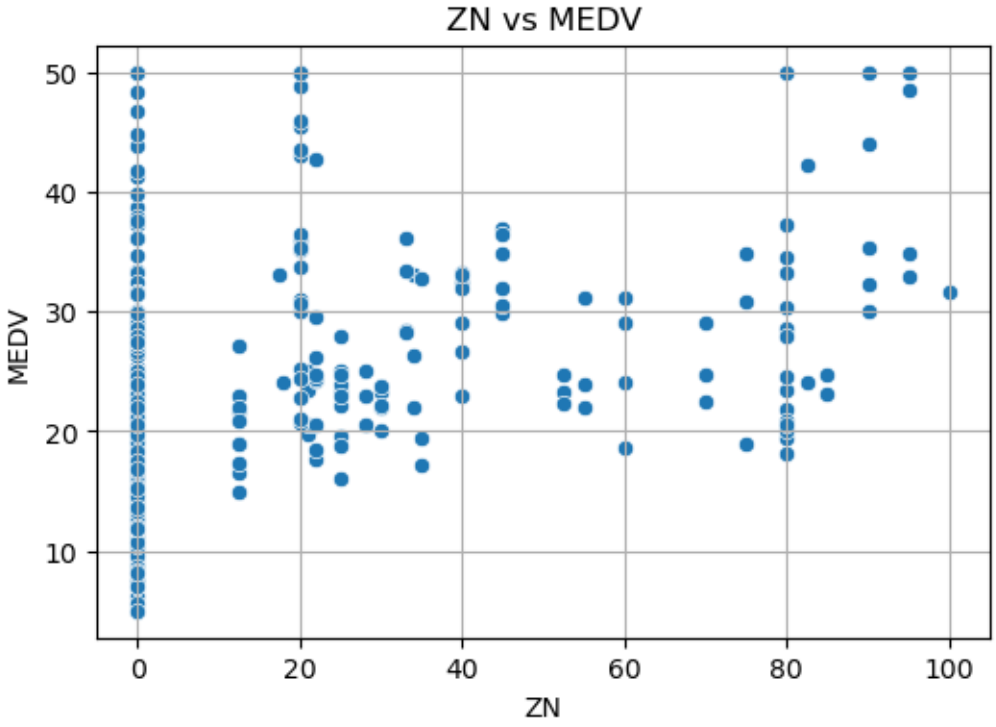
- Histogram reszt wykazuje asymetrię prawostronną (skośność dodatnia) – najwięcej reszt jest skupionych wokół 0, ale część ma duże dodatnie wartości odstające.

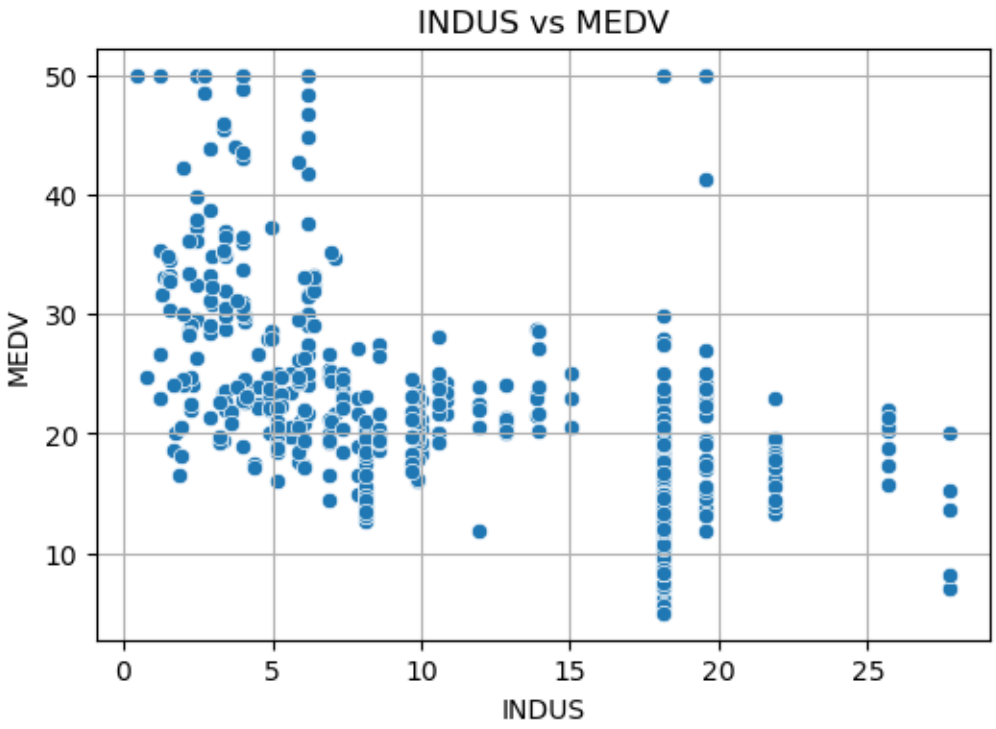
Sprawdzenie liniowości między predyktorami,a zmienną objaśnianą

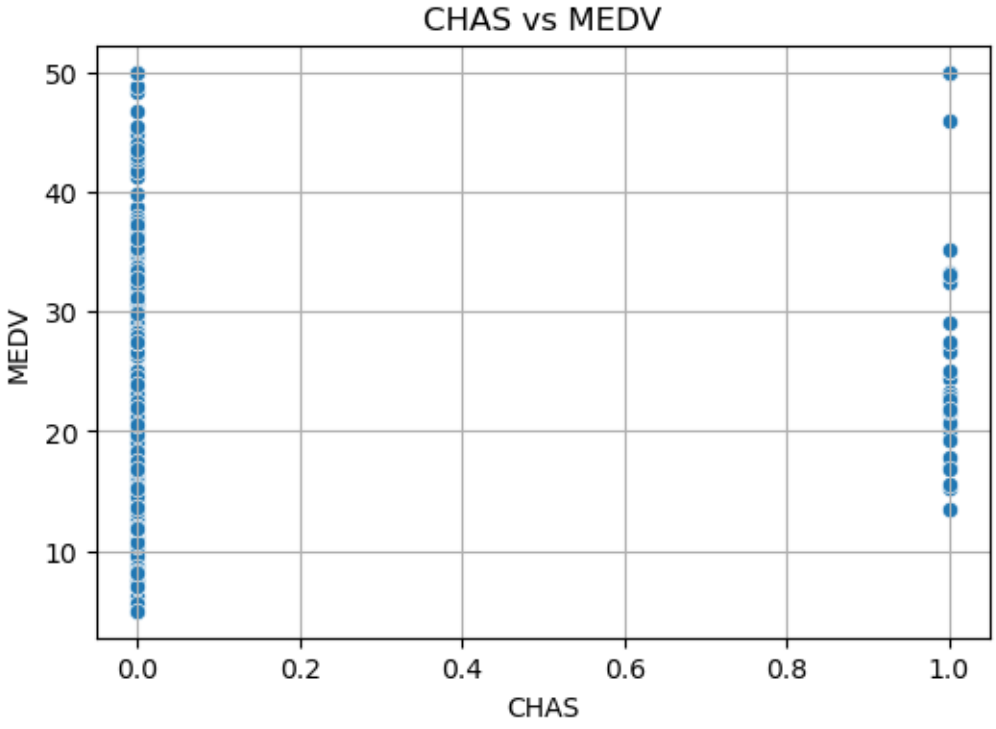
[19]:
```
features = dataset.drop('MEDV', axis=1)

for feature in features:
    plt.figure(figsize=(6, 4))
    sns.scatterplot(x=dataset[feature], y=dataset['MEDV'])
    plt.title(f'{feature} vs MEDV')
    plt.xlabel(feature)
    plt.ylabel('MEDV')
```

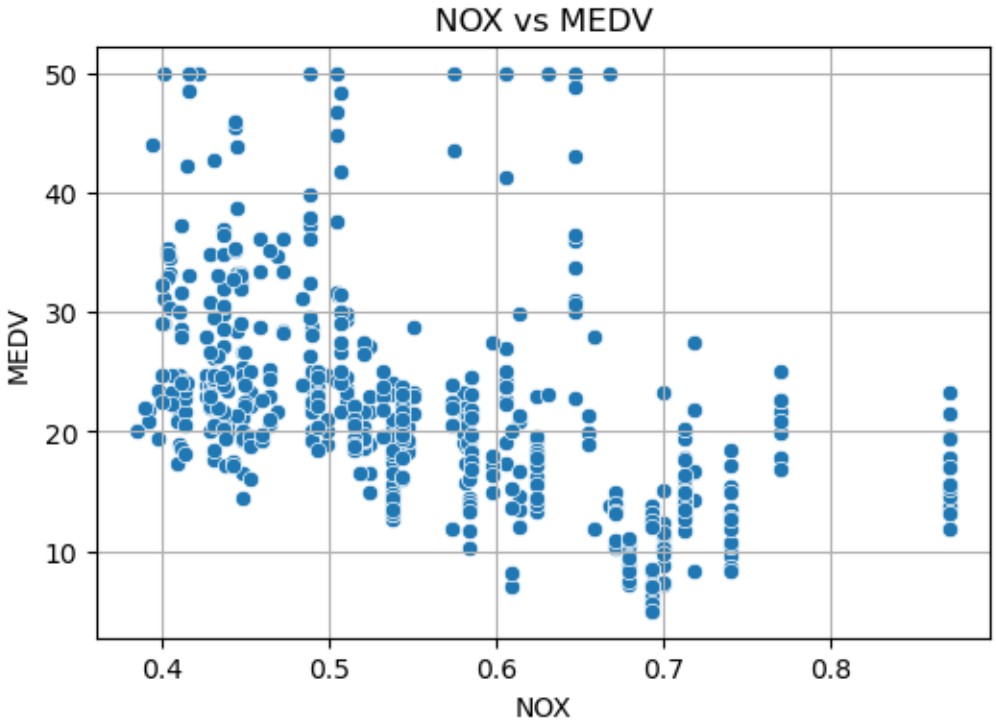
```
    plt.grid(True)
    plt.show()
```

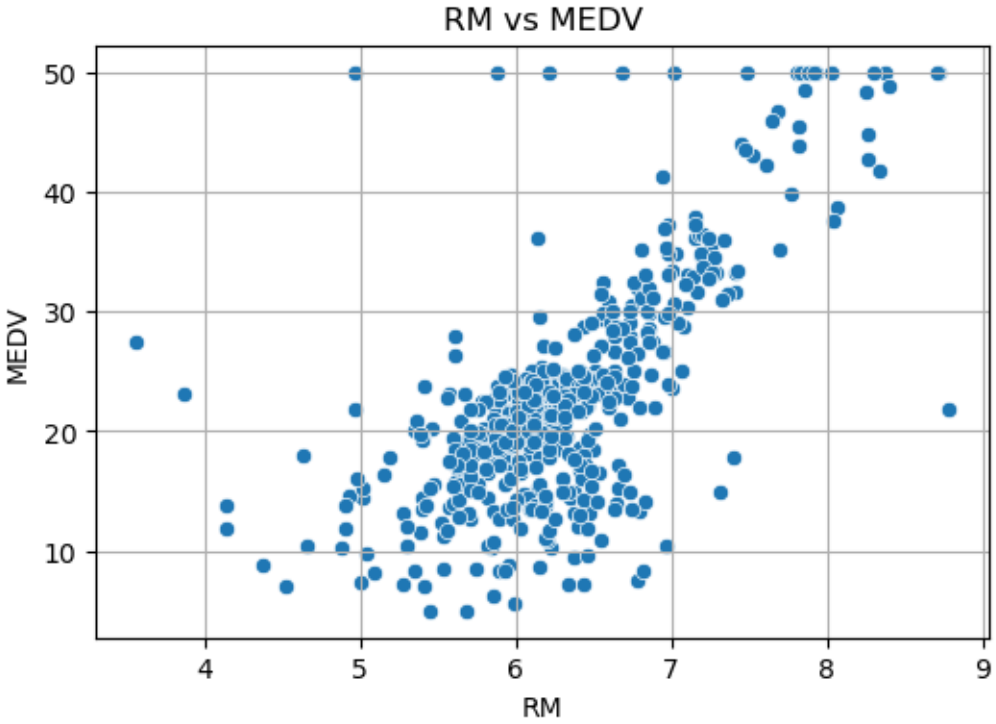


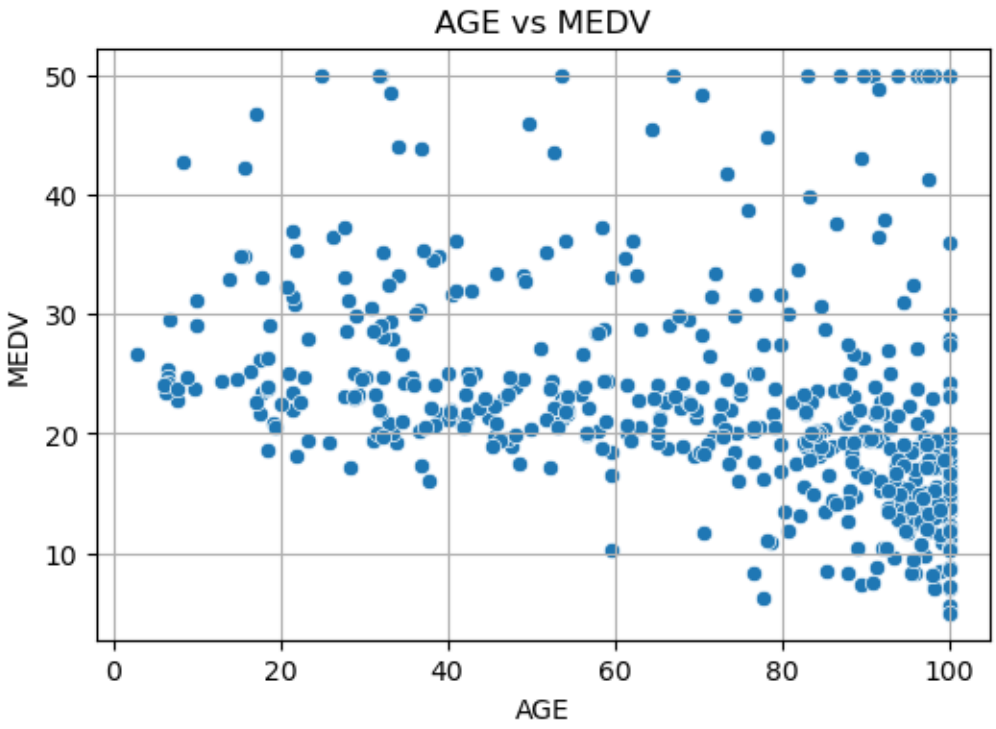


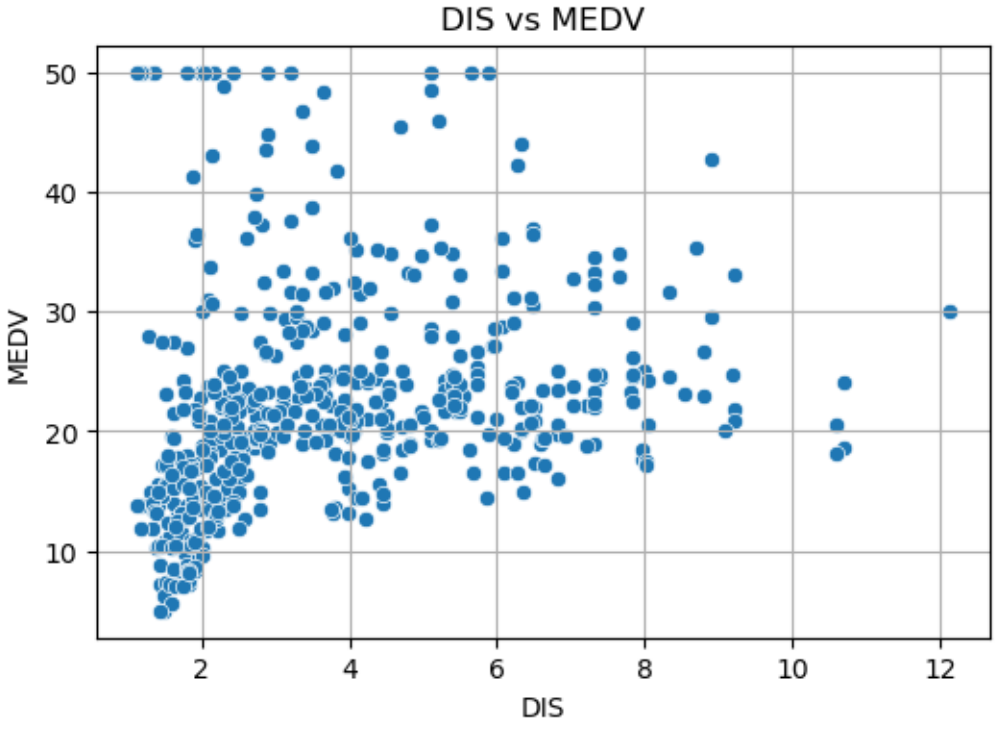

INDUS vs MEDV
MEDV
INDUS
50
40
30
20
10
0
5
10
15
20
25

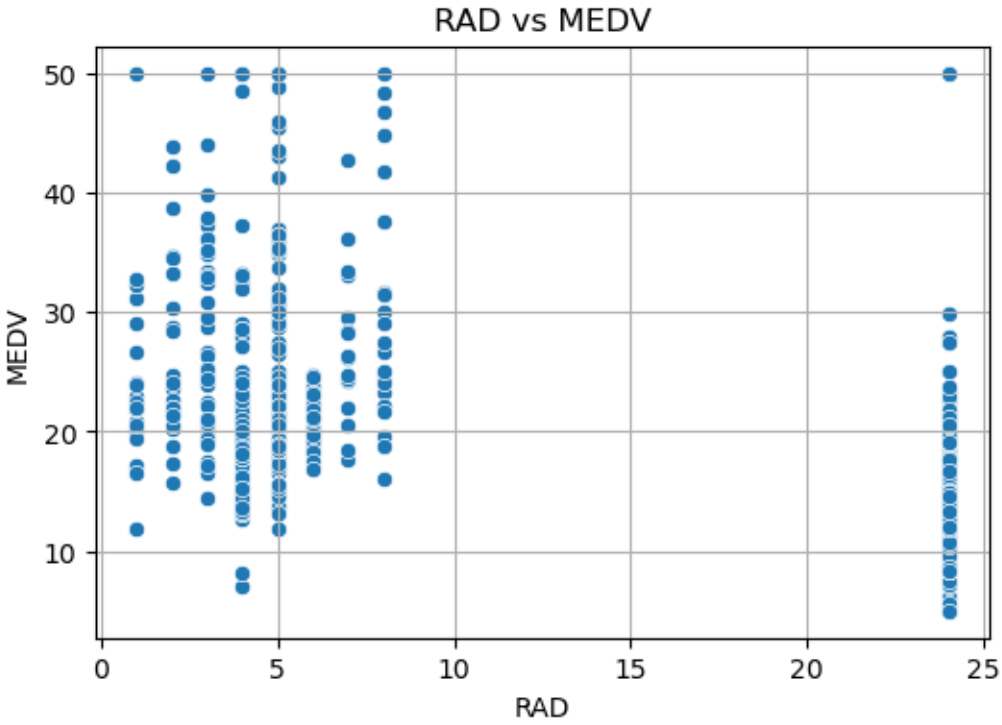

CHAS vs MEDV
MEDV
CHAS
50
40
30
20
10
0.0
0.2
0.4
0.6
0.8
1.0

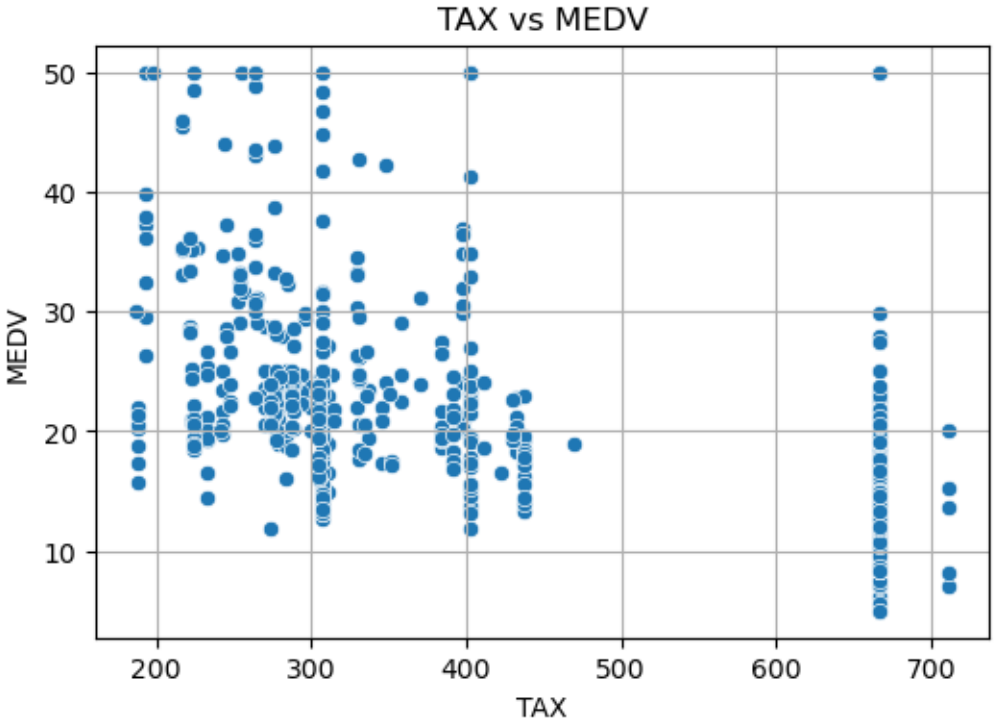

NOX vs MEDV
MEDV
NOX
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8

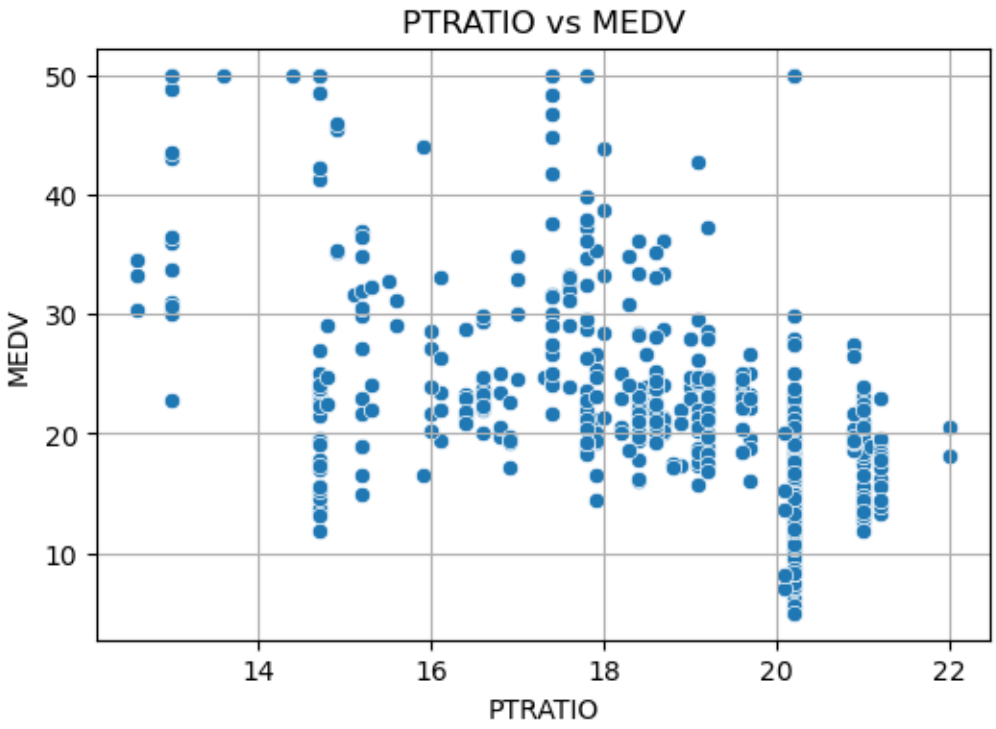

RM vs MEDV
MEDV
RM
50
40
30
20
10
4
5
6
7
8
9

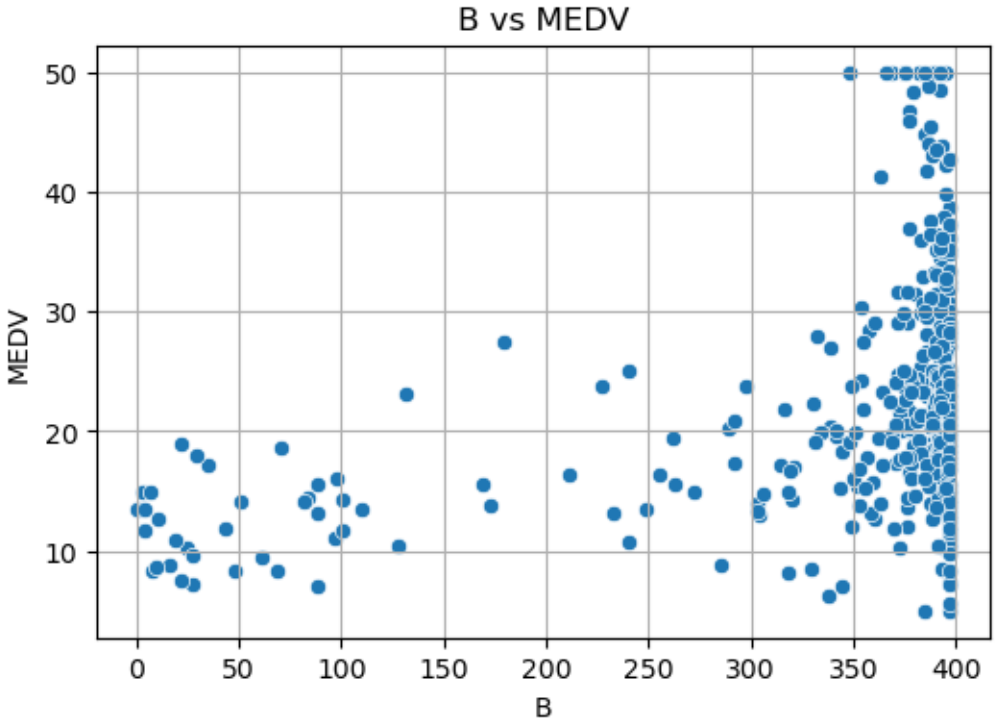

AGE vs MEDV
MEDV
AGE
50
40
30
20
10
0
20
40
60
80
100

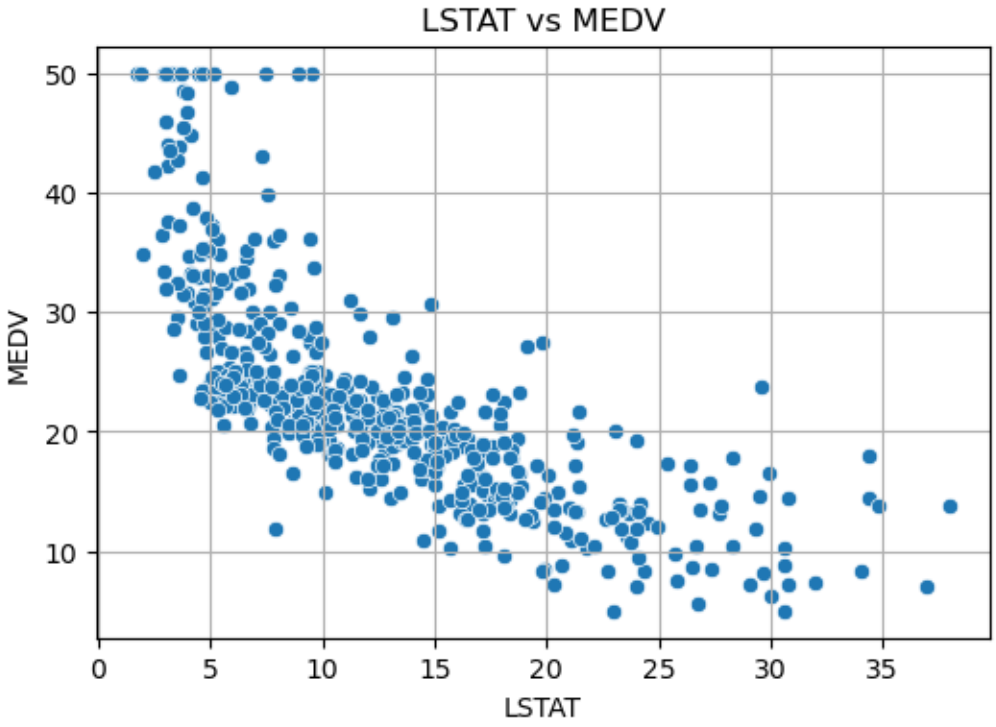

DIS vs MEDV
MEDV
DIS
50
40
30
20
10
2
4
6
8
10
12


RAD vs MEDV
MEDV
RAD
50
40
30
20
10
0
5
10
15
20
25


TAX vs MEDV
MEDV
TAX
50
40
30
20
10
200
300
400
500
600
700


PTRATIO vs MEDV
MEDV
PTRATIO
50
40
30
20
10
14
16
18
20
22


B vs MEDV
MEDV
B
50
40
30
20
10
0
50
100
150
200
250
300
350
400



Najsilniejsze pozytywne korelacje: - `RM` – średnia liczba pokoi na mieszkanie; silnie pozytywnie skorelowana z ceną. Więcej pokoi -> wyższa cena.

Najsilniejsze negatywne korelacje: - `LSTAT` – procent populacji o niższym statusie społecznym; najsilniejsza odwrotna zależność – im wyższy LSTAT, tym niższa cena.

[ ]: