Pobranie danych z Boston Housing Dataset i przypisanie nazwy kolumn

In [2]:
```
import pandas as pd

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

Out[2]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PIRATIO	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	39
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	39
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	39
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	39
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	39
...	...	...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	39
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	39
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	39
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	39
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	39

506 rows × 14 columns

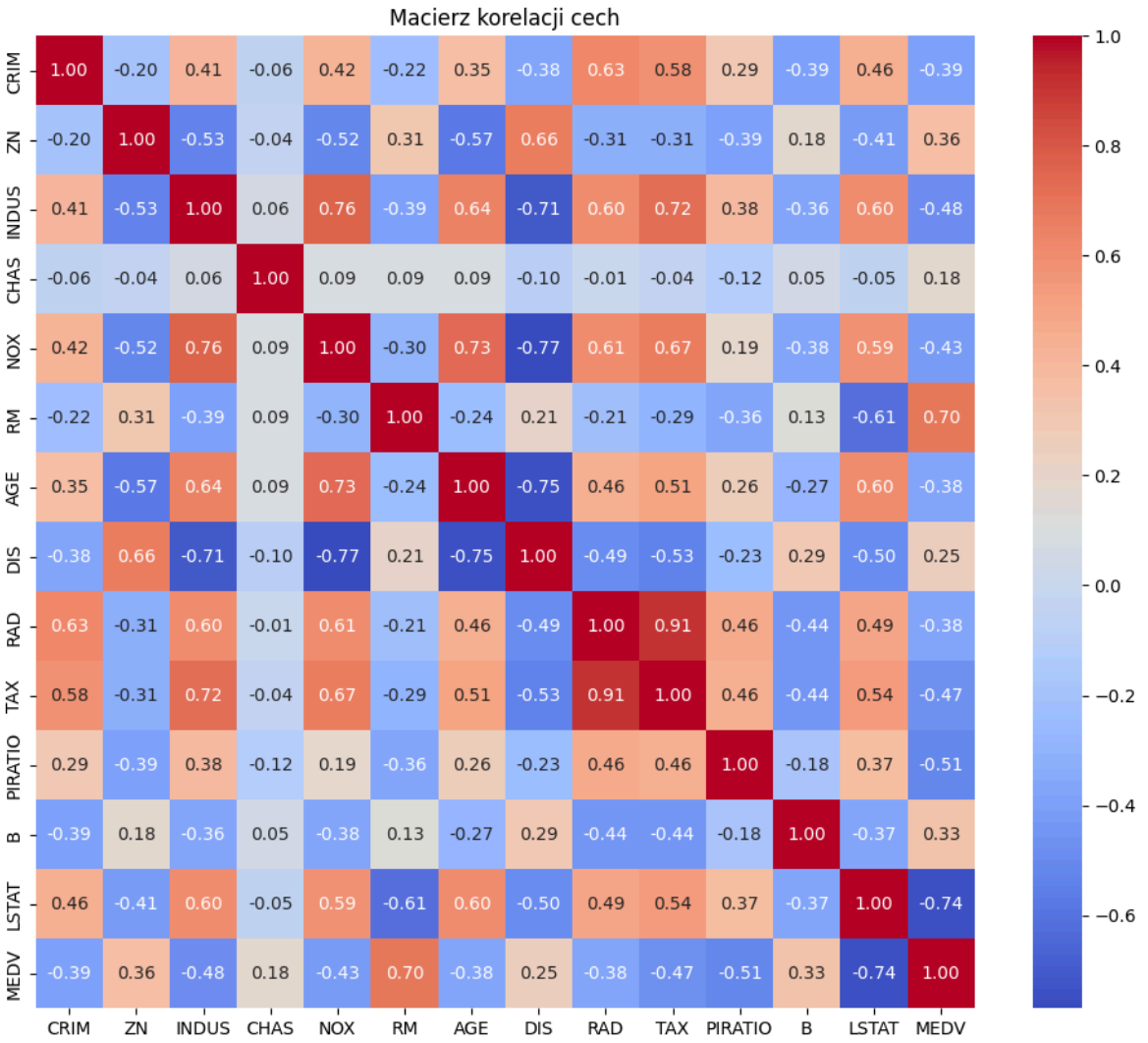
Podstawowa Exploratory Data Analysis (EDA)

In [14]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Podstawowe statystyki
print(dataset.describe())

# Korelacje
plt.figure(figsize=(12, 10))
sns.heatmap(dataset.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Macierz korelacji cech')
plt.show()
```

	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	PIRATIO	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



Najsilniejszą dodatnią korelację zaobserwowano między zmiennymi TAX i RAD, co może wskazywać na współliniowość.

Silne ujemne korelacje między DIS a INDUS, NOX i AGE, oraz LSTAT a MEDV, również mogą świadczyć o potencjalnej współliniowości między tymi zmiennymi.

CHAS jako zmienna najsłabiej skorelowana z innymi nie powoduje współliniowości, ale jej wpływ predykcyjny może być ograniczony.

Podział zbioru danych na zbiór treningowy i testowy

In [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_
```

Modele - regresja liniowa i XGBoost

In [10]:
```
from sklearn.linear_model import LinearRegression
import xgboost as xgb
```

```
# Regresja liniowa
lr = LinearRegression()
lr.fit(X_train, y_train)

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

Out[10]:

XGBRegressor

```
XGBRegressor(alpha=10, base_score=None, booster=None, callbacks=Non
e,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.3, device=None, early_stopping_round
s=None,
             enable_categorical=False, eval_metric=None, feature_typ
es=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.1, max_bi
```

Optymalne hiperparametry dla modelu XGBoost - GridSearchCV

In [11]:

```
from sklearn.model_selection import GridSearchCV

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

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

grid = GridSearchCV(estimator=xg_reg, param_grid=params, cv=5, scoring='neg_mean
grid.fit(X_train, y_train)

print("Best score: ", grid.best_score_)
print("Best params: ", grid.best_params_)
```

```
Best score:  -13.743169435966848
Best params:  {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 200}
```

Porównanie wyników różnych modeli na zbiorze testowym

In [12]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Regresja liniowa
y_pred_lr = lr.predict(X_test)

print("Linear Regression:")
print("  MSE: ", mean_squared_error(y_test, y_pred_lr))
print("  MAE: ", mean_absolute_error(y_test, y_pred_lr))
print("  R2:  ", r2_score(y_test, y_pred_lr))
```

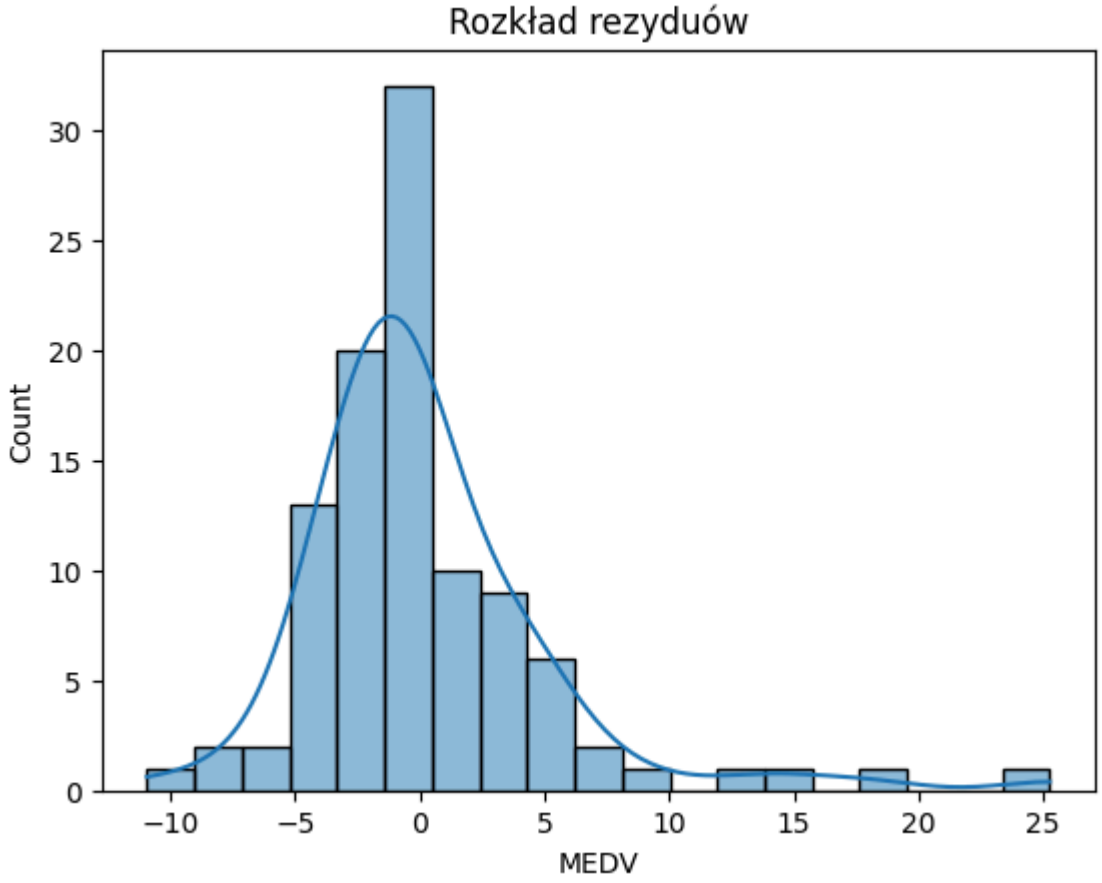
```
# XGBoost
xg_reg.fit(X_train, y_train)
y_pred_xg = xg_reg.predict(X_test)

print("XGBoost:")
print("  MSE: ", mean_squared_error(y_test, y_pred_xg))
print("  MAE: ", mean_absolute_error(y_test, y_pred_xg))
print("  R2:  ", r2_score(y_test, y_pred_xg))
```

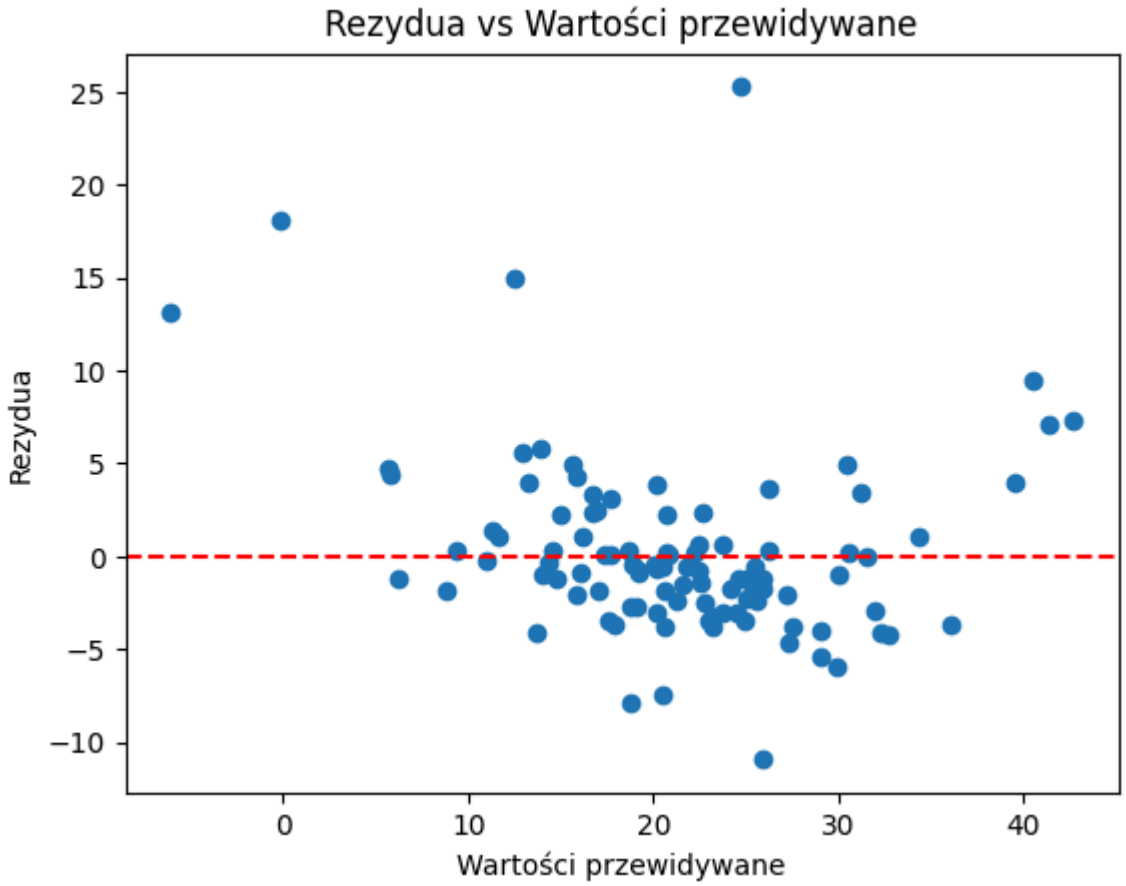
```
Linear Regression:
  MSE:  24.291119474973478
  MAE:  3.189091965887837
  R2:   0.6687594935356326
XGBoost:
  MSE:  27.48358210839519
  MAE:  3.3904440028994687
  R2:   0.6252261792043328
```

Sprawdzenie założeń dla regresji liniowej

In [17]:

```
# Sprawdzenie reszt
import scipy.stats as stats

residuals = y_test - y_pred_lr

# Histogram reszt
sns.histplot(residuals, kde=True)
plt.title("Rozkład rezyduów")
plt.show()

# Homoscedastyczność
plt.scatter(y_pred_lr, residuals)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel("Wartości przewidywane")
plt.ylabel("Rezydua")
plt.title("Rezydua vs Wartości przewidywane")
plt.show()
```





1. Rozkład reszt (górny wykres)

Rozkład jest w miarę normalny, co jest dobrą oznaką, ale nie idealny - jest lekko skośny w prawo.

2. Reszty vs Wartości przewidywane (dolny wykres)

Model nie spełnia w pełni założenia homoskedastyczności – wariancja reszt nie jest stała. Nie widać też wyraźnego wzoru nieliniowego.