Politechnika Krakowska im. Tadeusza Kościuszki
Wydział Inżynierii Elektrycznej i Komputerowej
Informatyka w Inżynierii Komputerowej

Studia niestacjonarne. Semestr 8

Temat: Uczenie liniowej hipotezy dla problemu przewidywania cen nieruchomości (Housing Data)

Przedmiot: Sztuczna Inteligencja

Prowadzący: mgr inż. Kazimierz Kiełkowicz

Autorzy:

Karolina Kilian

Artur Kotarba

1. Cel projektu

Celem projektu jest opracowanie modelu predykcyjnego, który wykorzystuje liniową regresję do przewidywania cen nieruchomości na podstawie zestawu danych dotyczących cech domów (Housing Data). Model ten ma na celu analizę zależności między zmiennymi opisującymi nieruchomości a ich cenami, co pozwoli na dokładne prognozowanie wartości rynkowych domów. Przewidywania te mogą być użyteczne dla potencjalnych nabywców, sprzedawców oraz analityków rynku nieruchomości.

2. Wstęp teoretyczny

Wstęp do problemu przewidywania cen nieruchomości

Przewidywanie cen nieruchomości to klasyczny problem w dziedzinie uczenia maszynowego, który polega na prognozowaniu wartości domów na podstawie różnych cech, takich jak lokalizacja, liczba pokoi, powierzchnia użytkowa, rok budowy itp. Jest to przykład problemu regresji, gdzie celem jest przewidywanie ciągłej wartości (ceny) na podstawie zestawu cech (zmiennych niezależnych).

Narzędzie: Regresja liniowa

Regresja liniowa to jedno z najprostszych i najczęściej stosowanych narzędzi w statystyce i uczeniu maszynowym do modelowania zależności między zmiennymi. Model regresji liniowej zakłada, że istnieje liniowa zależność między zmiennymi niezależnymi (cechami) a zmienną zależną (ceną nieruchomości).

Model regresji liniowej

Model regresji liniowej można przedstawić za pomocą równania:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta nxn + \epsilon$$

gdzie:

- y to zmienna zależna (cena nieruchomości),
- $x1, x2, \ldots, xn$ to zmienne niezależne (cechy nieruchomości),
- $\beta 0$ to wyraz wolny (intercept),
- $\beta 1, \beta 2, \ldots, \beta n$ to współczynniki regresji (parametry modelu),
- ϵ to składnik losowy (błąd).

Metoda uczenia: Metoda najmniejszych kwadratów

Najczęściej stosowaną metodą uczenia modelu regresji liniowej jest metoda najmniejszych kwadratów (OLS - Ordinary Least Squares). Celem OLS jest znalezienie takich wartości współczynników $\beta 0, \beta 1, \ldots, \beta n,$ które minimalizują sumę kwadratów odchyleń przewidywanych wartości od rzeczywistych wartości zmiennej zależnej.

Funkcja kosztu

Funkcja kosztu w metodzie najmniejszych kwadratów jest dana wzorem:

$$J(\beta) = \sum i = 1m(h\beta(x(i)) - y(i))2$$

gdzie:

- m to liczba obserwacji,
- $h\beta(x(i))$ to przewidywana wartość dla i-tej obserwacji,
- $y(i)$ to rzeczywista wartość dla i-tej obserwacji.

Minimalizując tę funkcję kosztu, otrzymujemy wartości parametrów β, które najlepiej dopasowują model do danych.

3. Kod oraz opis

3.1 Opis zmiennych w California Housing Dataset:

MedInc: Średni dochód gospodarstwa domowego w danej dzielnicy

HouseAge: Średni wiek domów w danej dzielnicy

AveRooms: Średnia liczba pokoi w domu

AveBedrms: Średnia liczba sypialni w domu

Population: Populacja w danej dzielnicy

AveOccup: Średnia liczba mieszkańców na gospodarstwo domowe

Latitude: Szerokość geograficzna danej dzielnicy

Longitude: Długość geograficzna danej dzielnicy

MedHouseVal: Mediana wartości domu w danej dzielnicy (w setkach tysięcy USD)

```
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge, Lasso
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# Załadowanie danych
california = fetch_california_housing()
df = pd.DataFrame(california.data, columns=california.feature_names)
df['MedHouseVal'] = california.target

# Podgląd danych
print(df.head())
print(df.describe())
```

```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude  MedHouseVal
0    -122.23        4.526
1    -122.22        3.585
2    -122.24        3.521
3    -122.25        3.413
4    -122.25        3.422
             MedInc      HouseAge      AveRooms     AveBedrms    Population  \
count  20640.000000  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.870671     28.639486      5.429000      1.096675   1425.476744
std        1.899822     12.585558      2.474173      0.473911   1132.462122
min        0.499900      1.000000      0.846154      0.333333      3.000000
25%        2.563400     18.000000      4.440716      1.006079    787.000000
50%        3.534800     29.000000      5.229129      1.048780   1166.000000
75%        4.743250     37.000000      6.052381      1.099526   1725.000000
max       15.000100     52.000000    141.909091     34.066667  35682.000000

            AveOccup      Latitude     Longitude   MedHouseVal
count  20640.000000  20640.000000  20640.000000  20640.000000
mean       3.070655     35.631861   -119.569704      2.068558
std       10.386050      2.135952      2.003532      1.153956
min        0.692308     32.540000   -124.350000      0.149990
25%        2.429741     33.930000   -121.800000      1.196000
50%        2.818116     34.260000   -118.490000      1.797000
75%        3.282261     37.710000   -118.010000      2.647250
max     1243.333333     41.950000   -114.310000      5.000010
```

3.2 Analiza brakujących wartości

Weryfikuje, czy w danych występują brakujące wartości, co jest kluczowe przed przeprowadzeniem dalszych analiz.

```
# Sprawdzenie brakujących wartości
missing_values = df.isnull().sum()
print("Missing values in each column:\n", missing_values)
```

```
Missing values in each column:
 MedInc         0
HouseAge        0
AveRooms        0
AveBedrms       0
Population      0
AveOccup        0
Latitude        0
Longitude       0
MedHouseVal     0
dtype: int64
```

3.3 Analiza eksploracyjna

Analiza eksploracyjna z wykorzystaniem macierzy korelacji to proces badania zależności między różnymi zmiennymi w zbiorze danych przy użyciu macierzy korelacji. Korelacja mierzy stopień, w jakim dwie zmienne są ze sobą powiązane. W kontekście macierzy korelacji, każdy element macierzy reprezentuje współczynnik korelacji między dwiema zmiennymi. Wartości korelacji zawierają się w przedziale od -1 do 1:

- 1 oznacza doskonałą dodatnią korelację: gdy jedna zmienna rośnie, druga również rośnie.

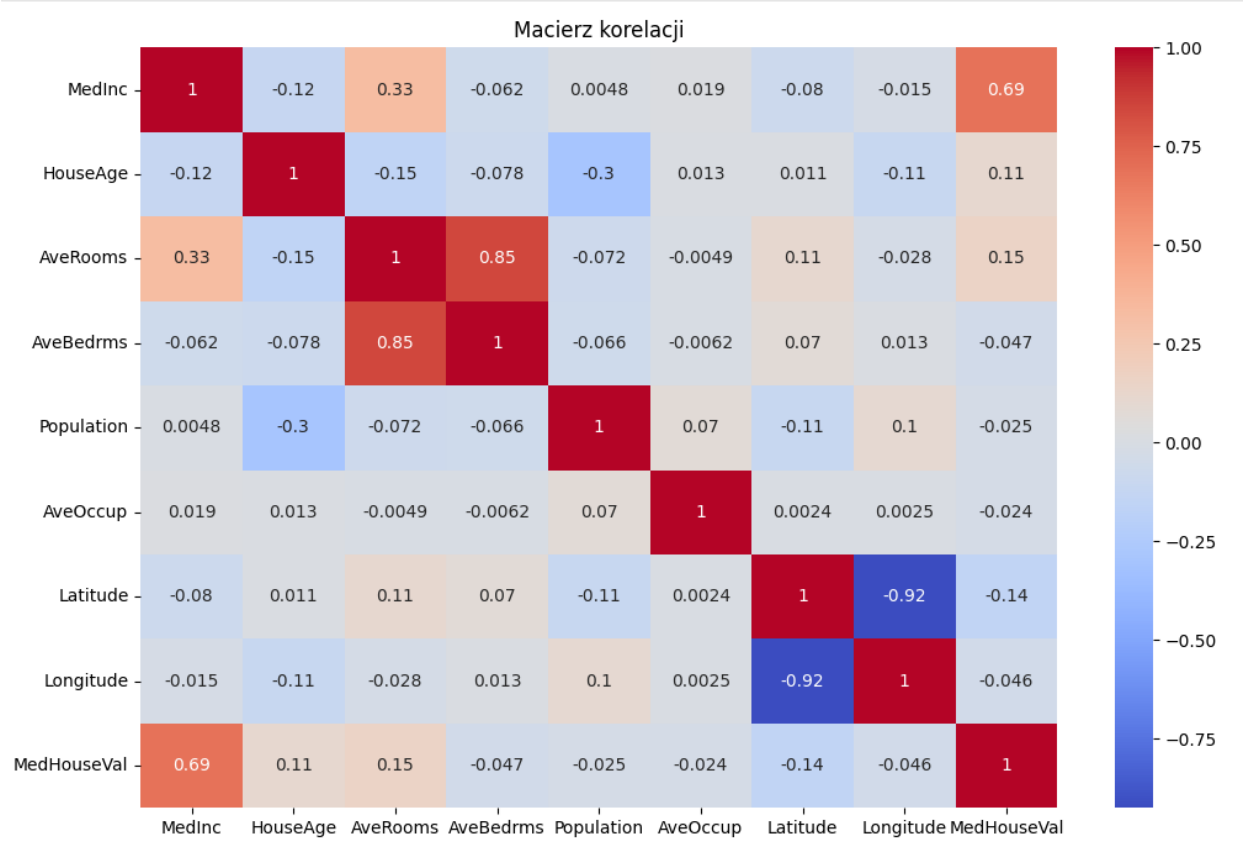
- -1 oznacza doskonałą ujemną korelację: gdy jedna zmienna rośnie, druga maleje.

- 0 oznacza brak korelacji: zmienne są niezależne.

Tworzenie macierzy korelacji i jej wizualizacja

Macierz korelacji jest kwadratową macierzą, gdzie każdy element [i, j] przedstawia korelację między zmienną i i zmienną j.

```
4. # Analiza eksploracyjna
5. plt.figure(figsize=(12, 8))
6. sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
7. plt.title('Macierz korelacji')
8. plt.show()
```



3.4 Standaryzacja danych

```
# Standaryzacja danych (cechy niezależne)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(df.drop('MedHouseVal', axis=1))
```

Standaryzacja danych to proces transformacji zmiennych w zbiorze danych, aby miały średnią równą 0 i odchylenie standardowe równe 1. Jest to często stosowana technika w przetwarzaniu danych, szczególnie w kontekście algorytmów uczenia maszynowego, które mogą być wrażliwe na skalę danych. Standaryzacja jest zalecana, gdy dane mają różne jednostki miary i planujemy używać modeli opartych na odległościach lub gradientach.

Cele i zalety standaryzacji danych:

- Ujednolicenie skali zmiennych: standaryzacja umożliwia porównywanie zmiennych, które pierwotnie miały różne jednostki i zakresy wartości.
- Poprawa wydajności algorytmów uczenia maszynowego: niektóre algorytmy, takie jak regresja liniowa, k-NN, SVM czy sieci neuronowe, działają lepiej, gdy zmienne mają zbliżone skale. Standaryzacja pomaga w zbieżności algorytmów optymalizacyjnych i może prowadzić do lepszych wyników modelu.
- Zapobieganie dominacji zmiennych o większej skali: Zmienne o większej skali mogą dominować w obliczeniach odległości czy wyznaczaniu granic decyzyjnych. Standaryzacja zapewnia, że wszystkie zmienne mają równy wpływ.

Standaryzacja danych powinna być wykonana po wstępnej analizie eksploracyjnej, a nie na samym początku, z kilku powodów:

- **Zrozumienie oryginalnych danych:**

Przeprowadzając wstępną analizę eksploracyjną na oryginalnych danych, można lepiej zrozumieć zakresy wartości, rozkład danych oraz potencjalne problemy, takie jak brakujące wartości. Analiza oryginalnych danych pozwala również zidentyfikować cechy, które mogą wymagać specjalnego traktowania lub dalszej inżynierii cech.

- **Detekcja i obsługa brakujących wartości:**

Przed standaryzacją należy sprawdzić i ewentualnie obsłużyć brakujące wartości, aby nie wprowadzać błędów do procesu standaryzacji. Standaryzacja danych z brakującymi wartościami może prowadzić do nieprawidłowych wyników.

- **Dokładność interpretacji:**

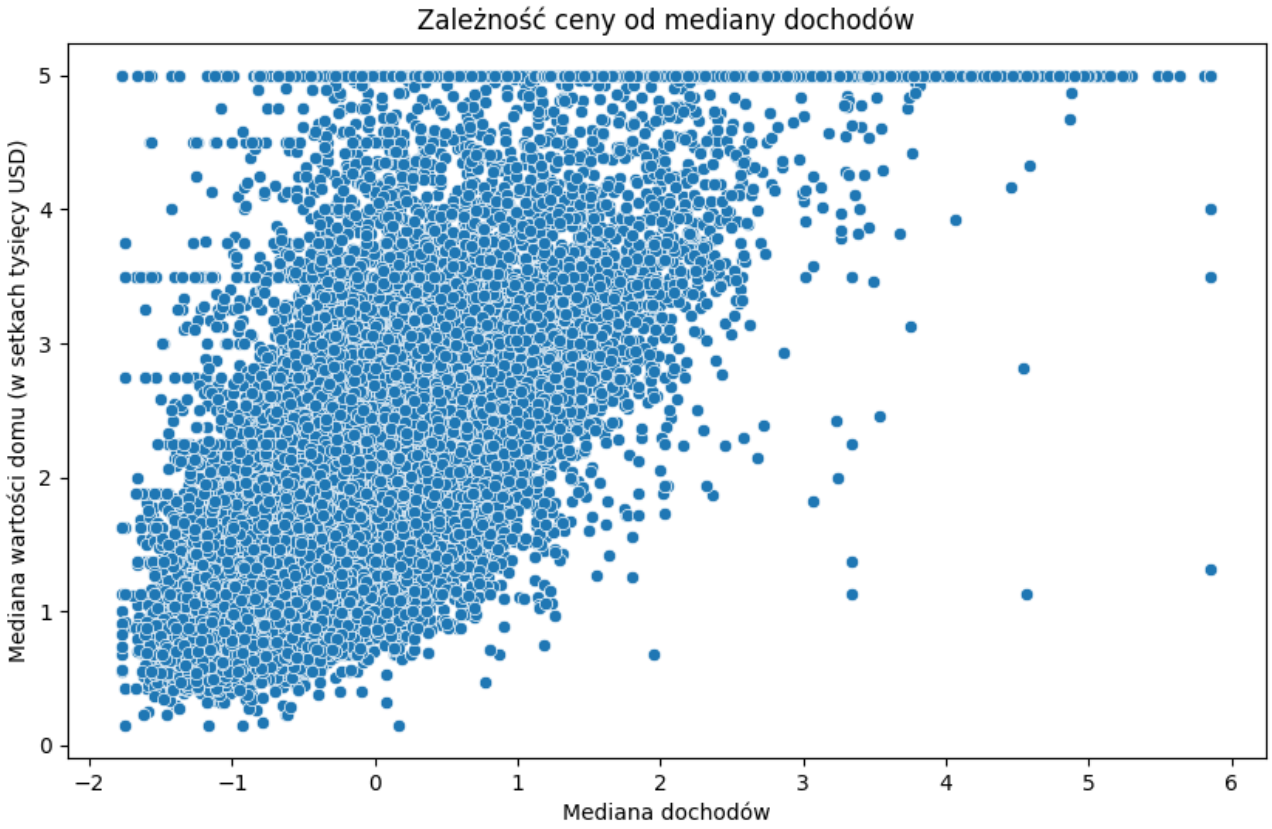
Standaryzacja zmienia skale cech, co może utrudnić interpretację surowych wartości. Przeprowadzając analizę eksploracyjną na oryginalnych danych, można lepiej zrozumieć, jak oryginalne wartości wpływają na zależności między cechami.

```
# Tworzenie nowej DataFrame z wystandaryzowanymi danymi
scaled_df = pd.DataFrame(scaled_features, columns=california.feature_names)
scaled_df['MedHouseVal'] = df['MedHouseVal']

# Podgląd wystandaryzowanych danych
print(scaled_df.head())
```

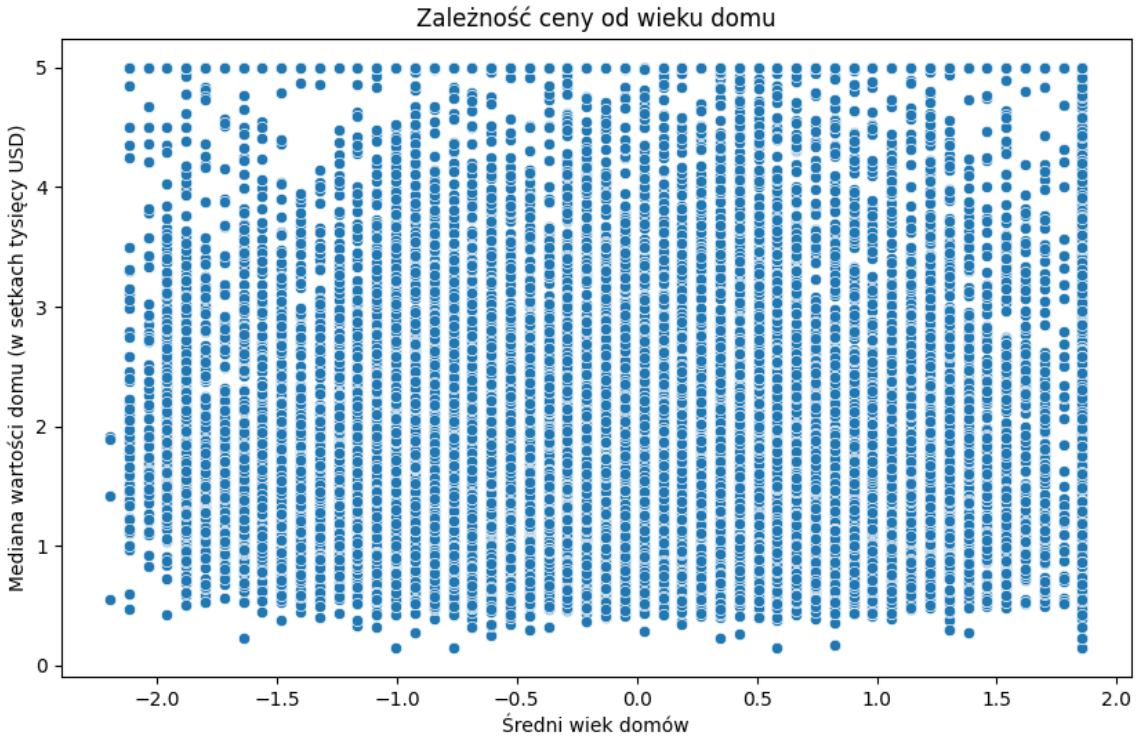
```
      MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  2.344766  0.982143  0.628559  -0.153758   -0.974429 -0.049597  1.052548
1  2.332238 -0.607019  0.327041  -0.263336    0.861439 -0.092512  1.043185
2  1.782699  1.856182  1.155620  -0.049016   -0.820777 -0.025843  1.038503
3  0.932968  1.856182  0.156966  -0.049833   -0.766028 -0.050329  1.038503
4 -0.012881  1.856182  0.344711  -0.032906   -0.759847 -0.085616  1.038503

   Longitude  MedHouseVal
0  -1.327835        4.526
1  -1.322844        3.585
2  -1.332827        3.521
3  -1.337818        3.413
4  -1.337818        3.422
```

3.5 Wizualizacja danych

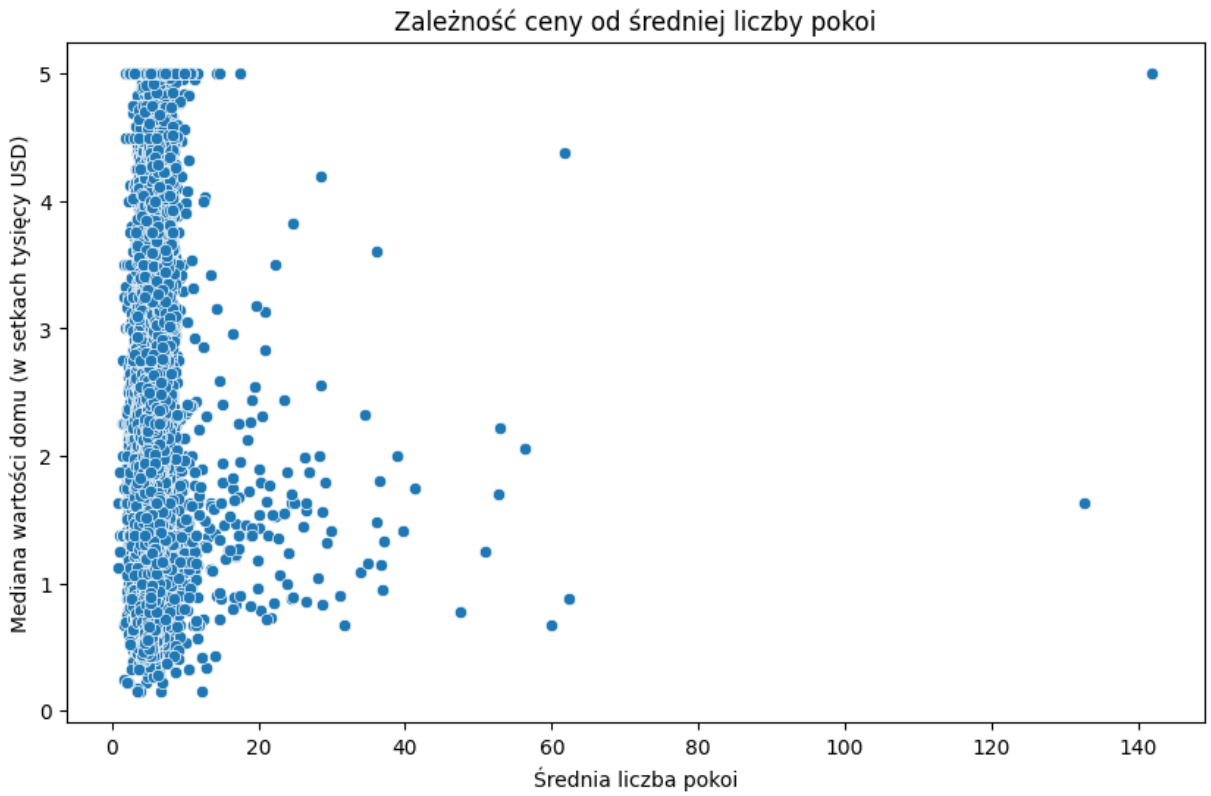
```
# Wizualizacja danych
scaled_df.hist(bins=50, figsize=(20, 15))
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='MedInc', y='MedHouseVal', data=scaled_df)
plt.title('Zależność ceny od mediany dochodów')
plt.xlabel('Mediana dochodów')
plt.ylabel('Mediana wartości domu (w setkach tysięcy USD)')
plt.show()
```



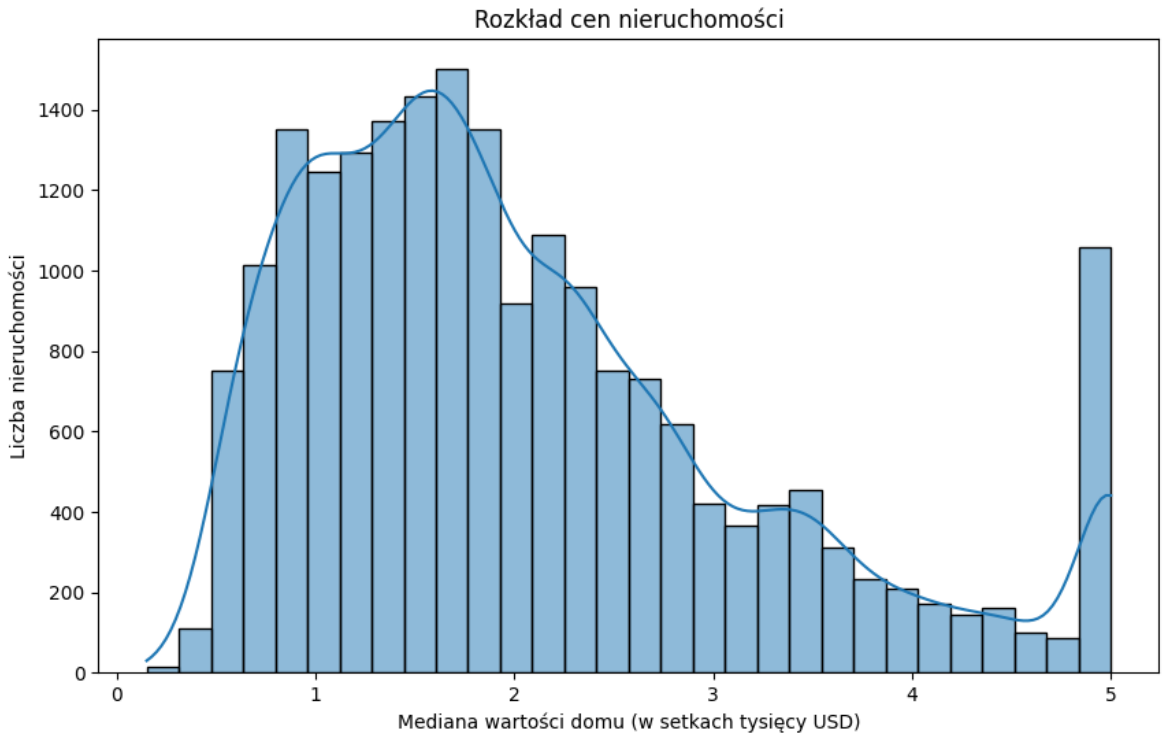

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='HouseAge', y='MedHouseVal', data=scaled_df)
plt.title('Zależność ceny od wieku domu')
plt.xlabel('Średni wiek domów')
plt.ylabel('Mediana wartości domu (w setkach tysięcy USD)')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='AveRooms', y='MedHouseVal', data=scaled_df)
plt.title('Zależność ceny od średniej liczby pokoi')
plt.xlabel('Średnia liczba pokoi')
plt.ylabel('Mediana wartości domu (w setkach tysięcy USD)')
plt.show()
```




```
plt.figure(figsize=(10, 6))
sns.histplot(scaled_df['MedHouseVal'], bins=30, kde=True)
plt.title('Rozkład cen nieruchomości')
plt.xlabel('Mediana wartości domu (w setkach tysięcy USD)')
plt.ylabel('Liczba nieruchomości')
plt.show()
```



3.6 Trenowanie modelu

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Podział danych na cechy (X) i zmienną docelową (y)
X = scaled_df.drop('MedHouseVal', axis=1)
y = scaled_df['MedHouseVal']

# Podział na zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Trenowanie modelu regresji liniowej
model = LinearRegression()
model.fit(X_train, y_train)

# Przewidywanie wartości na zbiorze testowym
y_pred = model.predict(X_test)
```

```python
# Obliczanie MSE
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error (MSE): {mse}")

# Obliczanie RMSE
rmse = np.sqrt(mse)
print(f"Root Mean Squared Error (RMSE): {rmse}")

# Obliczanie MAE
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error (MAE): {mae}")

# Obliczanie R²
r2 = r2_score(y_test, y_pred)
print(f"R-squared (R²): {r2}")
```

```
Mean Squared Error (MSE): 0.5558915986952441
Root Mean Squared Error (RMSE): 0.7455813830127762
Mean Absolute Error (MAE): 0.5332001304956565
R-squared (R²): 0.575787706032451
```

3.7 Analiza jakości modelu po zmianie parametrów konfiguracyjnych

Aby przeprowadzić analizę, jak zmienia się jakość modelu po zmianie parametrów konfiguracyjnych, musimy rozważyć różne aspekty konfiguracji modelu regresji liniowej oraz ewentualnie zastosować bardziej zaawansowane metody, takie jak regularyzacja, aby zoptymalizować parametry modelu.

1. Wybór modelu i jego podstawowe parametry

Dla modelu regresji liniowej w scikit-learn nie ma wielu parametrów do dostrojenia, ale możemy zastosować regularyzację, aby poprawić jakość modelu.

2. Dodanie regularyzacji

Regresja liniowa z regularyzacją (Lasso i Ridge) może poprawić wydajność modelu poprzez zmniejszenie nadmiernego dopasowania (overfitting).

- Lasso (L1 regularization): Dodaje karę za sumę wartości bezwzględnych współczynników regresji.
- Ridge (L2 regularization): Dodaje karę za sumę kwadratów współczynników regresji.

```
# Przeszukiwanie siatki dla Lasso
lasso_params = {'alpha': [0.01, 0.1, 1, 10, 100]}
lasso = Lasso()
lasso_grid = GridSearchCV(lasso, lasso_params, cv=5)
lasso_grid.fit(X_train, y_train)

# Przeszukiwanie siatki dla grid
ridge = Ridge()
param_grid = {'alpha': [0.01, 0.1, 1, 10, 100]}
ridge_grid = GridSearchCV(ridge, param_grid, cv=5,
scoring='neg_mean_squared_error')
ridge_grid.fit(X_train, y_train)

# Przeszukiwanie siatki dla ElasticNet
elasticnet_params = {'alpha': [0.01, 0.1, 1, 10, 100], 'l1_ratio': [0.1, 0.5,
0.7, 0.9, 1.0]}
elasticnet = ElasticNet()
elasticnet_grid = GridSearchCV(elasticnet, elasticnet_params, cv=5,
scoring='neg_mean_squared_error')
elasticnet_grid.fit(X_train, y_train)

def evaluate_model(model, X_train, y_train, X_test, y_test):
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"MSE: {mse:.4f}, RMSE: {rmse:.4f}, MAE: {mae:.4f}, R²: {r2:.4f}")

# Ocena najlepszego modelu Ridge
print("Ridge Regression:")
best_ridge = ridge_grid.best_estimator_
evaluate_model(best_ridge, X_train, y_train, X_test, y_test)

# Ocena najlepszego modelu Lasso
print("Lasso Regression:")
best_lasso = lasso_grid.best_estimator_
```

```
evaluate_model(best_lasso, X_train, y_train, X_test, y_test)

# Ocena najlepszego modelu ElasticNet
print("ElasticNet Regression:")
best_elasticnet = elasticnet_grid.best_estimator_
evaluate_model(best_elasticnet, X_train, y_train, X_test, y_test)
```

```
Ridge Regression:
MSE: 0.5559, RMSE: 0.7456, MAE: 0.5332, R²: 0.5758
Lasso Regression:
MSE: 0.5479, RMSE: 0.7402, MAE: 0.5355, R²: 0.5819
ElasticNet Regression:
MSE: 0.5515, RMSE: 0.7427, MAE: 0.5335, R²: 0.5791
```

Po uruchomieniu powyższego kodu otrzymujemy wyniki dla modeli Ridge, Lasso i ElasticNet. Wyniki te pomogą nam ocenić, czy regularyzacja poprawia jakość modelu w porównaniu do standardowej regresji liniowej.

Analiza wyników:

Wyniki dla Lasso Regression pokazują poprawę w porównaniu do podstawowego modelu regresji liniowej. RMSE i MAE są mniejsze, a R^2 jest wyższe, co wskazuje na lepsze dopasowanie modelu.

Ridge Regression w prównaniu do modelu bazowego daje bardzo zbliżone wyniki, oznacza to, że dane są dobrze przygotowane, a model regresji liniowej jest już optymalny dla tego zestawu danych. Regularizacja Ridge nie przynosi znaczących korzyści w tym przypadku. Ridge Regression wprowadziły regularizację, która zapobiega nadmiernemu dopasowaniu, ale jeśli dane nie mają dużej ilości szumu lub nie są skomplikowane, efekty regularizacji mogą być minimalne

Można również rozważyć inne metody regularizacji, takie jak ElasticNet, które łączą zarówno L1, jak i L2 regularizację. Jest przydatna, gdy mamy wiele cech i podejrzewamy, że niektóre z nich mogą być ze sobą skorelowane. Opis: ElasticNet wprowadza karę będącą kombinacją kar L1 i L2, co pozwala na jednoczesne

zmniejszanie wartości współczynników i wyzerowanie niektórych z nich. Wyniki dla ElasticNet pokazują poprawę w porównaniu do podstawowego modelu regresji liniowej.

4. Podsumowanie

Podstawowe statystyki opisowe: Pomagają zrozumieć rozkład każdej zmiennej i identyfikować potencjalne anomalie.

Analiza brakujących wartości: Weryfikuje, czy w danych występują brakujące wartości, co jest kluczowe przed przeprowadzeniem dalszych analiz.

Wizualizacja rozkładu zmiennych wejściowych: Pokazuje rozkład zmiennych w zestawie danych, co może pomóc zidentyfikować zmienne, które mogą wymagać transformacji.

Wizualizacja zależności między ceną a innymi zmiennymi: Pomaga zidentyfikować, które zmienne mają największy wpływ na wartość nieruchomości.

Wizualizacja rozkładu cen nieruchomości: Pokazuje, jak ceny są rozłożone w zestawie danych.

Wizualizacja korelacji między zmiennymi: Umożliwia identyfikację silnych zależności między zmiennymi, co jest przydatne przy budowie modeli predykcyjnych.

Optymalizacja parametrów: Dodanie regularyzacji (ElasticNety lub Lasso) pozwala na poprawę jakości modelu regresji liniowej. Lasso Regression wydaje się działać lepiej w tym przypadku, redukując błędy i zwiększając wartość R^2. W tym przypadku Ridge Regression nie przyniosła poprawy modelu bazowego.

Zmiana parametrów konfiguracyjnych i zastosowanie regularyzacji może znacząco poprawić jakość modelu regresji liniowej, co jest widoczne w niższych wartościach RMSE i MAE oraz wyższych wartościach R^2. Jednak zawsze warto przeprowadzić dokładne testy i porównać różne modele, aby znaleźć optymalne rozwiązanie dla danego problemu.