Statystyczne metody rozpoznawania obrazu

Zadanie 4

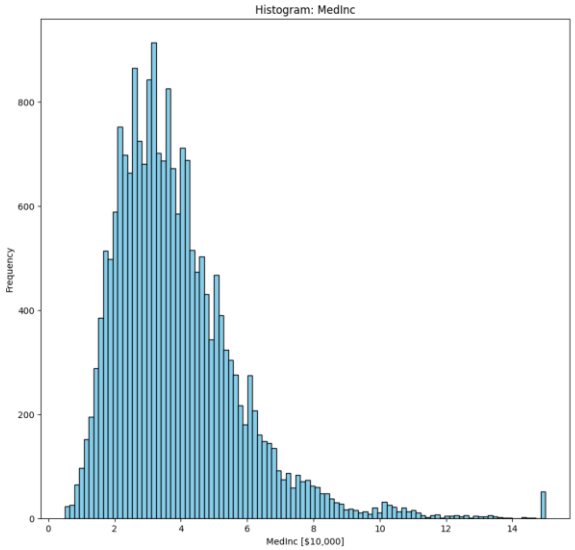
Bartłomiej Gorzela

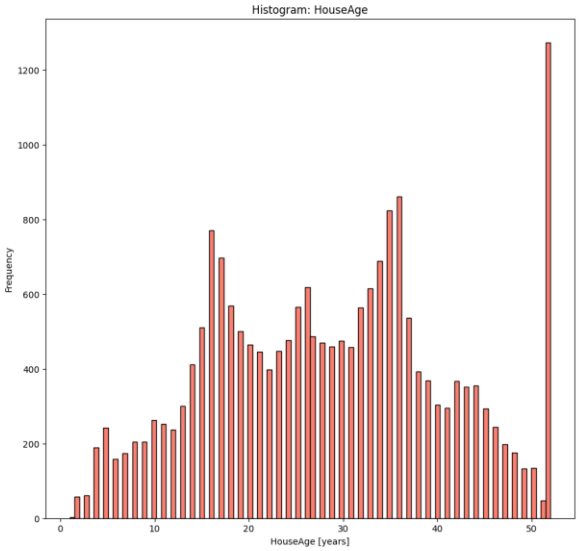
1. **Zidentyfikować wartości odstające w wybranym zbiorze danych.**

W celu wykonania zadania posłużyłem się przykładowym zbiorem danych ‘California Housing Dataset’ możliwym do zaimportowania bezpośrednio z biblioteki scikit-learn dostępnej w Python. Do identyfikacji wartości odstających zdecydowałem się na wybór dwóch atrybutów:

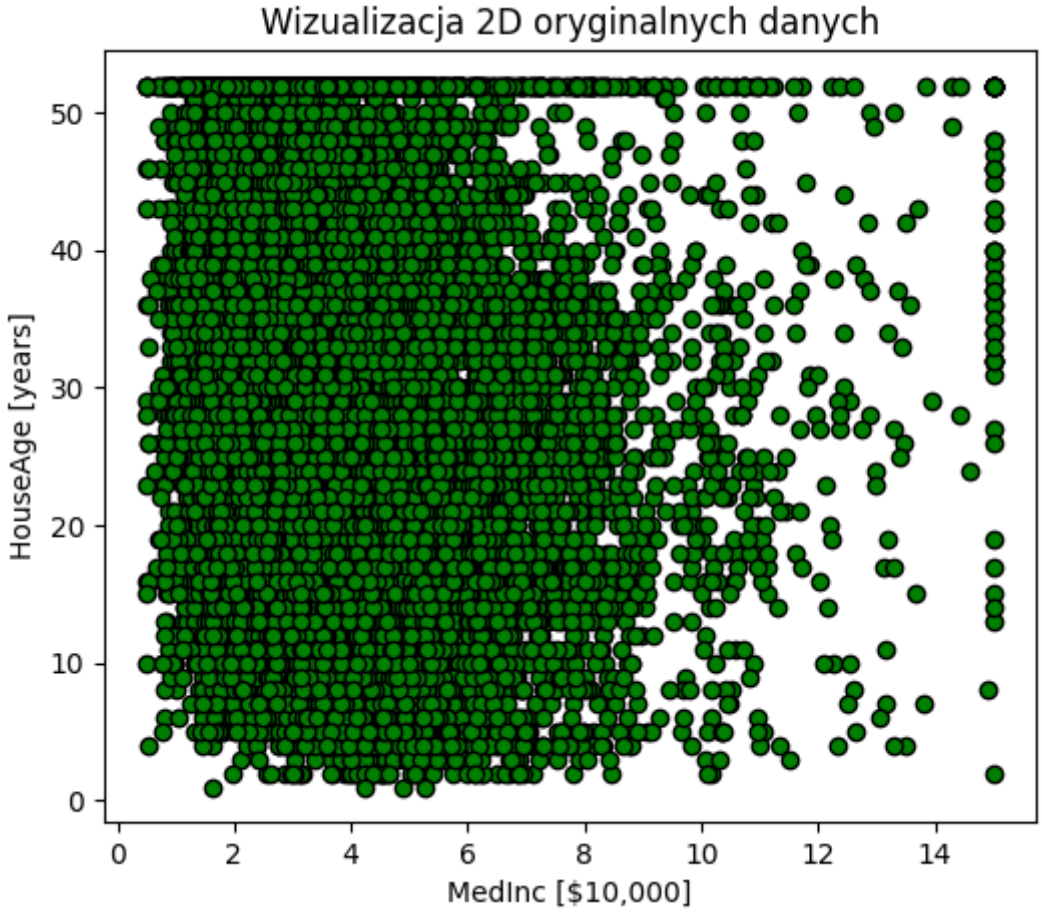
- *MedInc* (Median Income): przedstawia medianę dochodu gospodarstwa domowego.
- *HouseAge* (Median Age of Housing Units): średnia wieku wszystkich budynków mieszkalnych.

Wartości wybranych atrybutów zwizualizowałem za pomocą histogramów (argument ‘bins’ ustawiłem na wartość 100 w celu poprawy czytelności wizualizacji):





Oryginalne dane zwizualizowałem za pomocą wykresu 2D:

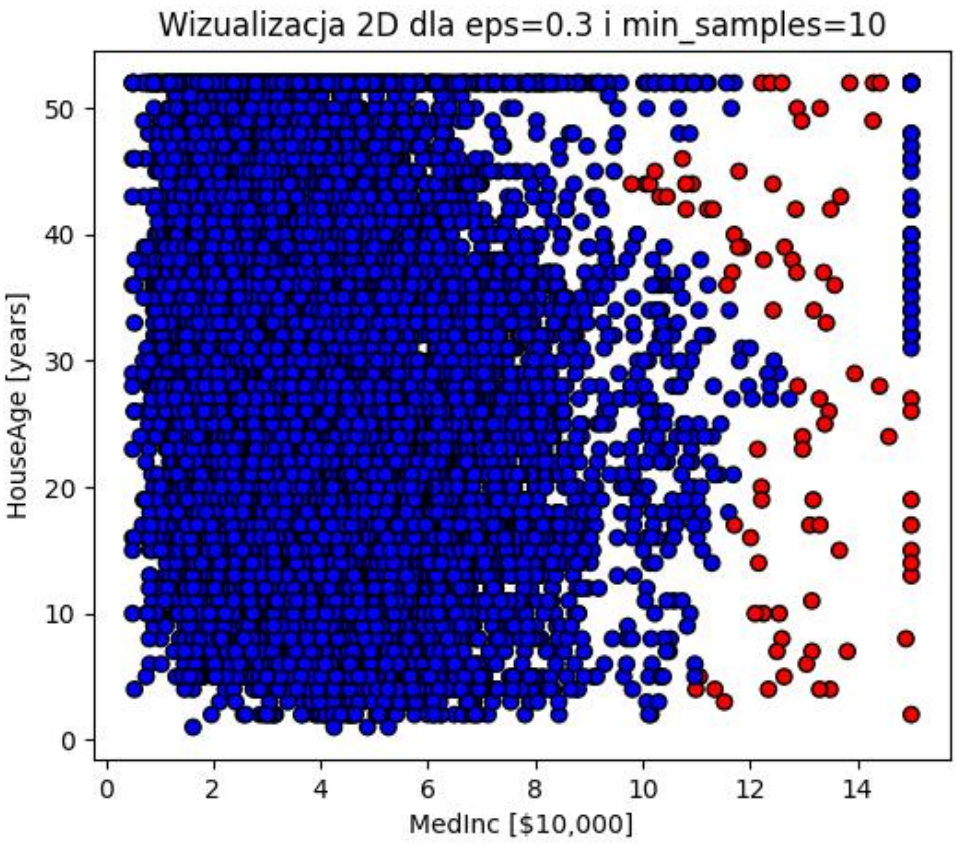


Kolejnym krokiem jaki wyknałem było zidentyfikowanie wartości odstających. Posłużyłem się w tym celu algorytmem DBSCAN (Density-Based Spatial Clustering of Applications with Noise), który umożliwia wykrywanie skupisk danych oraz jednoczesne oznaczanie punktów, które można uznać za wartości odstające. Algorytm jest dostępny w bibliotece scikit-learn.

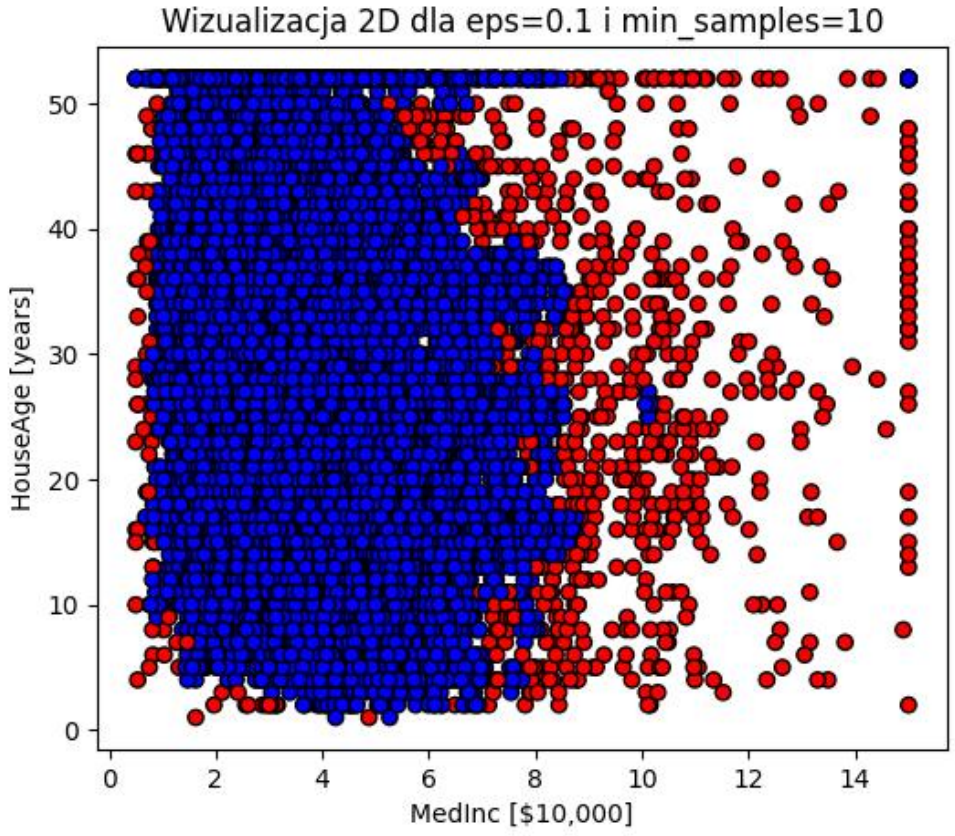
Algorytm wymagał przeprowadzenia standaryzacji danych, czego dokonałem przy użyciu obiektu StandardScaler(). Po przeskalowaniu danych, zastosowałem algorytm DBSCAN z parametrami: **eps=0.3** oraz **min_samples=10**. Oznacza to, że punkt jest uznany za rdzeniowy jeśli ma co najmniej 10 sąsiadów w promieniu 0.3 jednostki (w przestrzeni standaryzowanej).

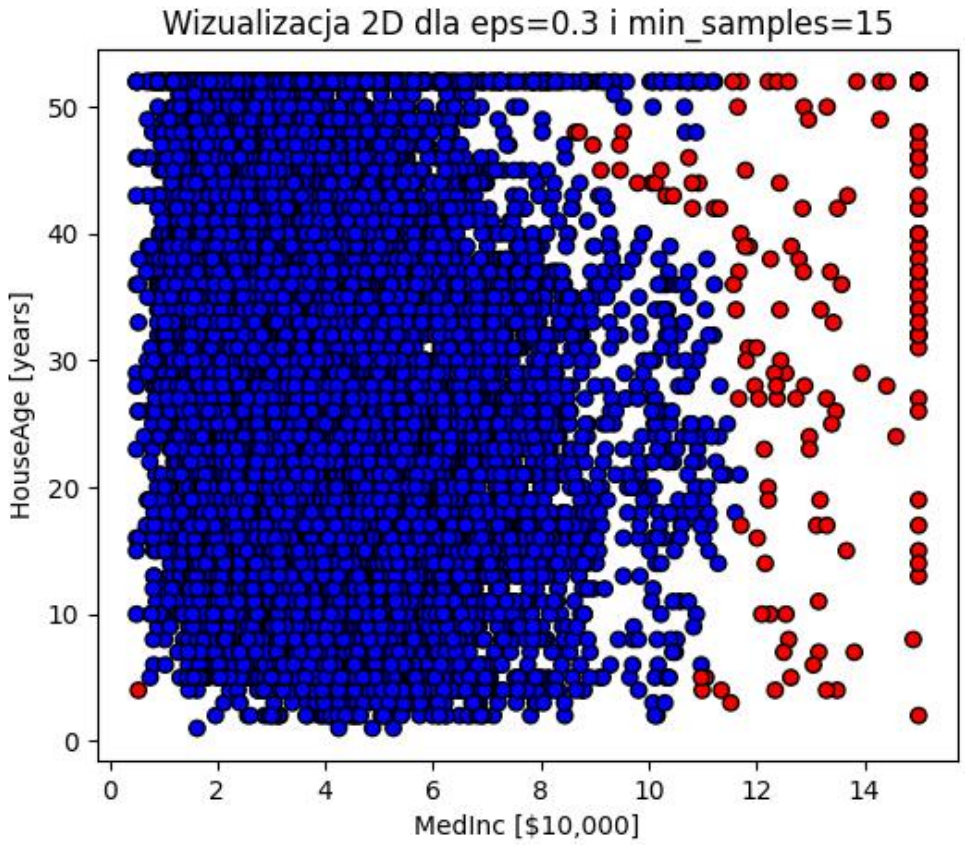
W wyniku działania DBSCAN każdemu punktowi została przypisana etykieta klastra (dbscan_label), przy czym wartość -1 oznacza punkt uznany za odstający.

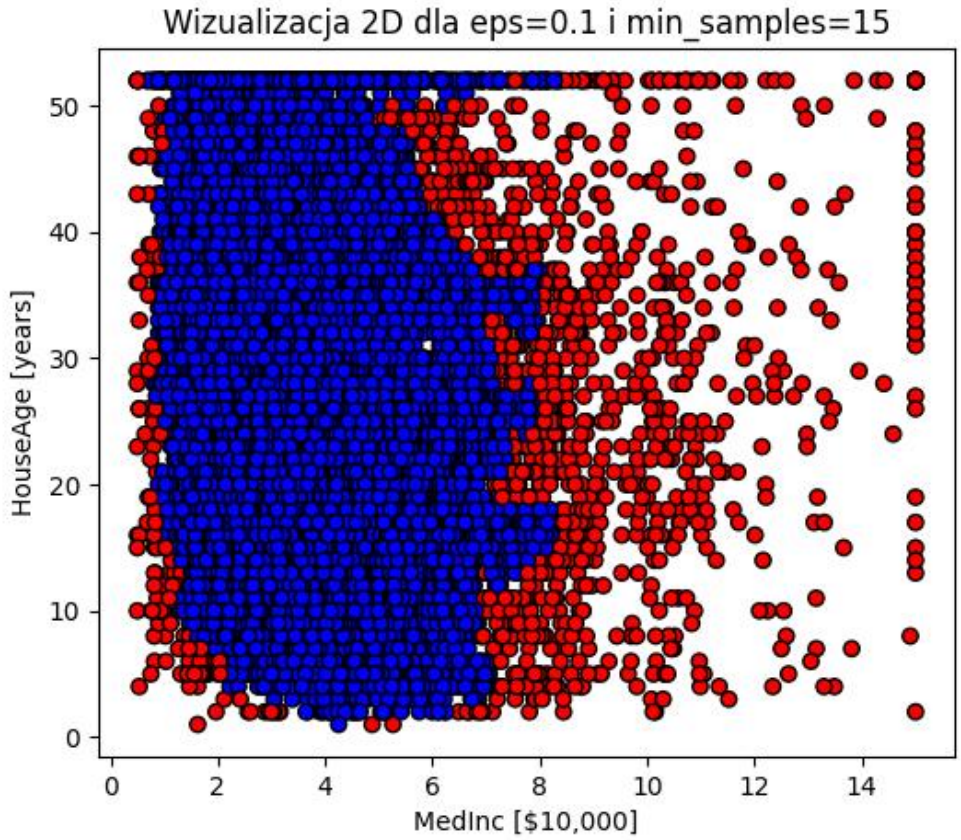
Poddane działaniu algorytmu atrybuty ponownie zwizualizowałem za pomocą wykresu 2D, zaznaczając kolorem czerwonym punkty uznane za wartości odstające.



Ten sam krok powtórzyłem dla modyfikując wartości ‘eps’ oraz ‘min_samples’. Wyniki znajdują się poniżej:




Wizualizacja 2D dla eps=0.3 i min_samples=15
HouseAge [years]
MedInc [$10,000]
0
10
20
30
40
50
0
2
4
6
8
10
12
14


Wizualizacja 2D dla eps=0.1 i min_samples=15
HouseAge [years]
MedInc [$10,000]
0
10
20
30
40
50
0
2
4
6
8
10
12
14

Kod źródłowy:

Bartłomiej_Gorzela_Zadanie_4.py U

StatisticalPatternRecognitionMethods > Task4 > Bartłomiej_Gorzela_Zadanie_4.py > ...

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

california = datasets.fetch_california_housing()
california_data = california.data
california_df = pd.DataFrame(california_data, columns=california.feature_names)

# Wizualiacja danych za pomocą histogramów
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.hist(california_df['MedInc'], bins=100, color='skyblue', edgecolor='black')
plt.title(f'Histogram: MedInc')
plt.xlabel('MedInc [$10,000]')
plt.ylabel('Frequency')
plt.subplot(1, 2, 2)
plt.hist(california_df['HouseAge'], bins=100, color='salmon', edgecolor='black')
plt.title(f'Histogram: HouseAge')
plt.xlabel('HouseAge [years]')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()

# Wizualizacja 2D oryginalnych danych
plt.figure(figsize=(6, 5))
plt.scatter(california_df['MedInc'], california_df['HouseAge'], color='green', edgecolor='k')
plt.xlabel('MedInc [$10,000]')
plt.ylabel('HouseAge [years]')
plt.title('Wizualizacja 2D oryginalnych danych')
plt.show()

# DBSCAN
attributes_for_scaling = california_df[['MedInc', 'HouseAge']]
scaler = StandardScaler()
attributes_scaled = scaler.fit_transform(attributes_for_scaling)
dbscan = DBSCAN(eps=0.3, min_samples=10)
# dbscan = DBSCAN(eps=0.1, min_samples=10)
# dbscan = DBSCAN(eps=0.3, min_samples=15)
# dbscan = DBSCAN(eps=0.1, min_samples=15)

labels = dbscan.fit_predict(attributes_scaled)
california_df['dbscan_label'] = labels

# Wizualizacja 2D z etykietami DBSCAN
plt.figure(figsize=(6, 5))
colors = ['red' if label == -1 else 'blue' for label in labels]
plt.scatter(california_df['MedInc'], california_df['HouseAge'], c=colors, edgecolor='k')
plt.xlabel('MedInc [$10,000]')
plt.ylabel('HouseAge [years]')
plt.title('Wizualizacja 2D dla eps=0.3 i min_samples=10')
# plt.title('Wizualizacja 2D dla eps=0.1 i min_samples=10')
# plt.title('Wizualizacja 2D dla eps=0.3 i min_samples=15')
# plt.title('Wizualizacja 2D dla eps=0.1 i min_samples=15')
plt.show()
```