Statystyczne metody rozpoznawania obrazu

Zadanie 3

Bartłomiej Gorzela

W celu wykonania zadania posłużyłem się przykładowym zbiorem danych ‘Iris’ możliwym do zaimportowania bezpośrednio z biblioteki scikit-learn dostępnej w Python.

1. **Podać zakres zmienności wartości atrybutów.**

Zakres zmienności wartości atrybutów obliczyłem jako różnicę pomiędzy maksymalną a minimalną z wartości dla każdego z atrybutów.

Kod źródłowy:

```
# 3.1 Podać zakres zmienności wartości atrybutów.
range = iris_df.max() - iris_df.min()
print("Zakres zmienności wartości atrybutów w zbiorze Iris:")
print(range)
```

Wynik:

```
Zakres zmienności wartości atrybutów w zbiorze Iris:
sepal length (cm)    3.6
sepal width (cm)     2.4
petal length (cm)    5.9
petal width (cm)     2.4
```

2. **Obliczyć średnią dla wartości atrybutów wybranego zbioru danych.**

Średnią wartość dla każnego z atrybutów obliczyłem za pomocą dostępnej funkcji mean().

Kod źródłowy:

```
# 3.2 Obliczyć średnią dla wartości atrybutów wybranego zbioru danych.
mean_values = iris_df.mean()
print("Średnie wartości atrybutów w zbiorze Iris:")
print(mean_values)
```

Wynik:

```
Srednie wartości atrybutów w zbiorze Iris:
sepal length (cm)    5.843333
sepal width (cm)     3.057333
petal length (cm)    3.758000
petal width (cm)     1.199333
```

3. Obliczyć odchylenie standardowe wartości atrybutów.

Odchylenie standardowe wartości każego z atrybutów policzyłem za pomocą dostępnej funkcji std().

Kod źródłowy:

```
# 3.3 Obliczyć odchylenie standardowe wartości atrybutów.
std_values = iris_df.std()
print("Odchylenia standardowe wartości atrybutów w zbiorze Iris:")
print(std_values)
```

Wynik:

```
Odchylenia standardowe wartości atrybutów w zbiorze Iris:
sepal length (cm)    0.828066
sepal width (cm)     0.435866
petal length (cm)    1.765298
petal width (cm)     0.762238
```

4. Przeskalować do pewnego przedziału. Wybraną metodą zwizualizować różnice między tak otrzymanymi wartościami cech a tymi przed przeskalowaniem.

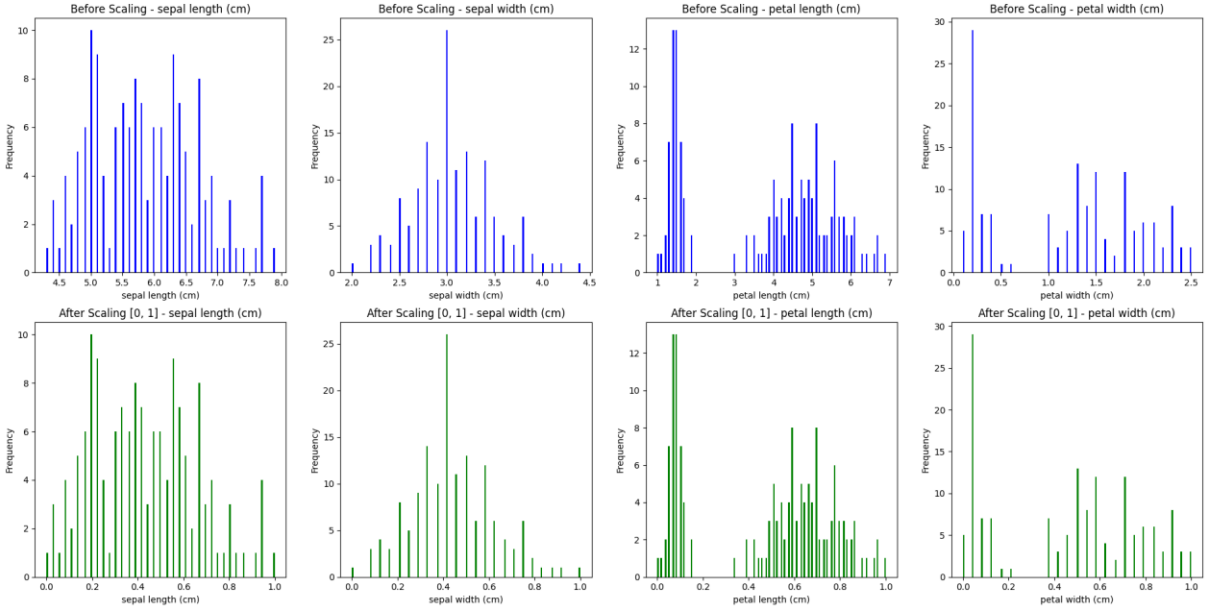
Wartości każdego z atrybutów przeskalowałem do przedziału [0, 1] przy pomocy obiektu MinMaxScaler. Różnicę przed i po skalowaniu przedstawiłem za pomocą histogramów.

Kod źródłowy:

```
# 3.4 Przeskalować do pewnego przedziału. Wybraną metodą zwizualizować różnice między tak otrzymanymi wartościami cech a tymi przed przeskalowaniem.
scaler = MinMaxScaler(feature_range=(0, 1))
iris_scaled = scaler.fit_transform(iris_df)
iris_scaled_df = pd.DataFrame(iris_scaled, columns=iris.feature_names)

fig, axes = plt.subplots(2, 4, figsize=(16, 10))
for i, feature in enumerate(iris.feature_names):
    # Przed przeskalowaniem
    axes[0, i].hist(iris_df[feature], bins=len(iris_df[feature]), color='blue', label=f'Before Scaling')
    axes[0, i].set_title(f'Before Scaling - {feature}')
    axes[0, i].set_xlabel(feature)
    axes[0, i].set_ylabel('Frequency')
    # Po przeskalowaniu
    axes[1, i].hist(iris_scaled_df[feature], bins=len(iris_scaled_df[feature]), color='green', label=f'After Scaling')
    axes[1, i].set_title(f'After Scaling [0, 1] - {feature}')
    axes[1, i].set_xlabel(feature)
    axes[1, i].set_ylabel('Frequency')
plt.tight_layout()
plt.show()
```

Wynik:



5. Wykonać standaryzację wartości atrybutów. Wybraną metodą zwizualizować różnice.

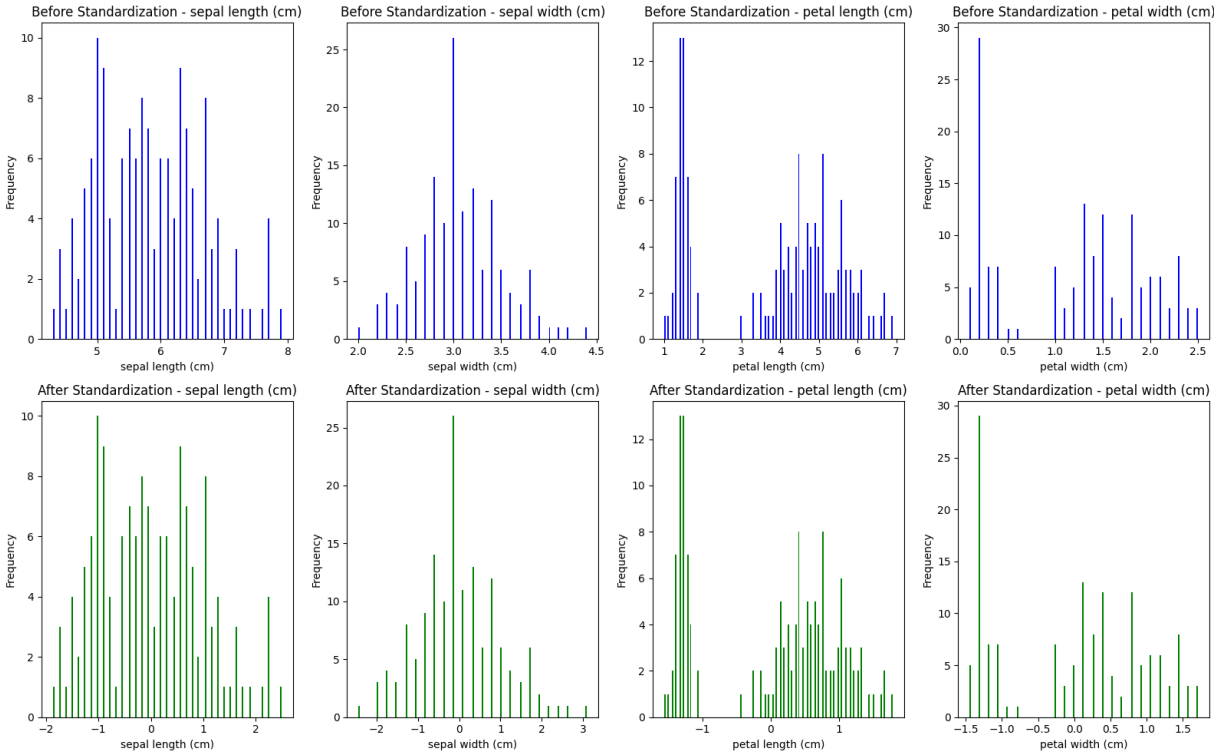
Standaryzacje dla każdego z atrybutów wykonałem przy pomocy obiektu StandardScaler. Różnicę przed i po standaryzacji przedstawiłem za pomocą histogramów.

Kod źródłowy:

```
# 3.5 Wykonać standaryzację wartości atrybutów. Wybraną metodą zwizualizować różnice.
scaler = StandardScaler()
iris_standardized = scaler.fit_transform(iris_df)
iris_standardized_df = pd.DataFrame(iris_standardized, columns=iris.feature_names)

fig, axes = plt.subplots(2, 4, figsize=(16, 10))
for i, feature in enumerate(iris.feature_names):
    # Przed standaryzacją
    axes[0, i].hist(iris_df[feature], bins=len(iris_df[feature]), color='blue', label=f'Before Standardization')
    axes[0, i].set_title(f'Before Standardization - {feature}')
    axes[0, i].set_xlabel(feature)
    axes[0, i].set_ylabel('Frequency')
    # Po standaryzacji
    axes[1, i].hist(iris_standardized_df[feature], bins=len(iris_standardized_df[feature]), color='green', label=f'After Standardization')
    axes[1, i].set_title(f'After Standardization - {feature}')
    axes[1, i].set_xlabel(feature)
    axes[1, i].set_ylabel('Frequency')
plt.tight_layout()
plt.show()
```

Wynik:


Before Standardization - sepal length (cm)
Before Standardization - sepal width (cm)
Before Standardization - petal length (cm)
Before Standardization - petal width (cm)
After Standardization - sepal length (cm)
After Standardization - sepal width (cm)
After Standardization - petal length (cm)
After Standardization - petal width (cm)
Frequency
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)

6. Przeprowadzić normalizację wartości atrybutów (przy użyciu różnych norm). Wybraną metodą zwizualizować różnice.

Normalizację wartości atrybutów wykonałem przy pomocy obiektu Normalizer. Posłużyłem się dwiema normami:

- L1: suma wartości bezwzględnych dla każdego wektora cech równa 1.

- L2: długość każdego wektora cech równa 1.

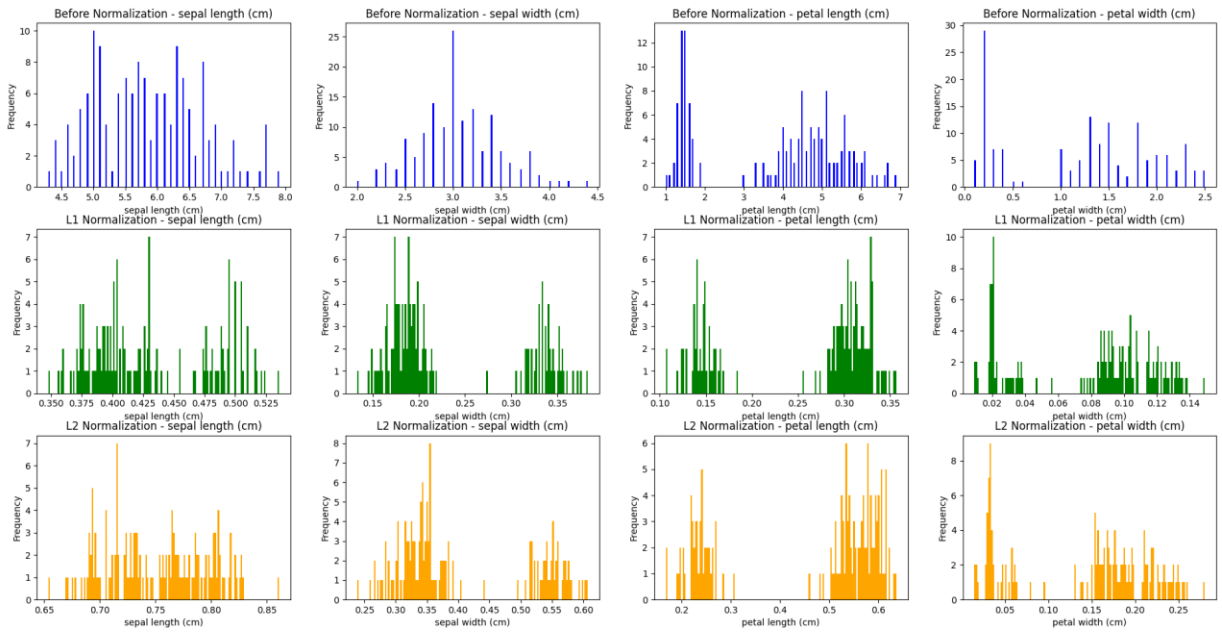
Różnicę przed i po normalizacji przedstawiłem za pomocą histogramów oraz wyświetliłem w konsoli.

Kod źródłowy:

```
# 3.6 Przeprowadzić normalizację wartości atrybutów (przy użyciu różnych norm). Wybraną metodą zwizualizować różnice.
normalizer_L1 = Normalizer(norm='l1')  # Norma L1
normalizer_L2 = Normalizer(norm='l2')  # Norma L2
iris_L1_normalized = normalizer_L1.fit_transform(iris_df)
iris_L2_normalized = normalizer_L2.fit_transform(iris_df)
iris_L1_df = pd.DataFrame(iris_L1_normalized, columns=iris.feature_names)
iris_L2_df = pd.DataFrame(iris_L2_normalized, columns=iris.feature_names)

fig, axes = plt.subplots(3, 4, figsize=(16, 12))
for i, feature in enumerate(iris.feature_names):
    # Przed normalizacją
    axes[0, i].hist(iris_df[feature], bins=len(iris_df[feature]), color='blue', label=f'Before Normalization')
    axes[0, i].set_title(f'Before Normalization - {feature}')
    axes[0, i].set_xlabel(feature)
    axes[0, i].set_ylabel('Frequency')
    # Norma L1
    axes[1, i].hist(iris_L1_df[feature], bins=len(iris_L1_df[feature]), color='green', label=f'L1 Normalization')
    axes[1, i].set_title(f'L1 Normalization - {feature}')
    axes[1, i].set_xlabel(feature)
    axes[1, i].set_ylabel('Frequency')
    # Norma L2
    axes[2, i].hist(iris_L2_df[feature], bins=len(iris_L2_df[feature]), color='orange', label=f'L2 Normalization')
    axes[2, i].set_title(f'L2 Normalization - {feature}')
    axes[2, i].set_xlabel(feature)
    axes[2, i].set_ylabel('Frequency')
plt.tight_layout()
plt.show()
print("Wartości atrybutów zbioru Iris przed normalizacją:")
print(iris_df)
print("Wartości atrybutów zbioru Iris po normalizacji normą L1:")
print(iris_L1_df)
print("Wartości atrybutów zbioru Iris po normalizacji normą L2:")
print(iris_L2_df)
```

Wynik:



```
Wartości atrybutów zbioru Iris przed normalizacją:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
Wartości atrybutów zbioru Iris po normalizacji normą L1:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0             0.500000          0.343137           0.137255          0.019608
1             0.515789          0.315789           0.147368          0.021053
2             0.500000          0.340426           0.138298          0.021277
3             0.489362          0.329787           0.159574          0.021277
4             0.490196          0.352941           0.137255          0.019608
..                 ...               ...                ...               ...
145           0.389535          0.174419           0.302326          0.133721
146           0.401274          0.159236           0.318471          0.121019
147           0.389222          0.179641           0.311377          0.119760
148           0.358382          0.196532           0.312139          0.132948
149           0.373418          0.189873           0.322785          0.113924

[150 rows x 4 columns]
Wartości atrybutów zbioru Iris po normalizacji normą L2:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0             0.803773          0.551609           0.220644          0.031521
1             0.828133          0.507020           0.236609          0.033801
2             0.805333          0.548312           0.222752          0.034269
3             0.800030          0.539151           0.260879          0.034784
4             0.790965          0.569495           0.221470          0.031639
..                 ...               ...                ...               ...
145           0.721557          0.323085           0.560015          0.247699
146           0.729654          0.289545           0.579090          0.220054
147           0.716539          0.330710           0.573231          0.220474
148           0.674671          0.369981           0.587616          0.250281
149           0.690259          0.350979           0.596665          0.210588

[150 rows x 4 columns]
```

7. Oczyścić dane (uzupełnić/usunąć brakujące wartości, usunąć duplikaty).

W celu wykonania zadania ponownie posłużyłem się zbiorem danych Iris, lecz tym razem wprowadziłem sztuczne modyfikacje, aby móc następnie oczyścić dane.

Pierwszą modyfikacją było ustawienie 4 pierwszych rekordów atrybutu ‘sepal length (cm)’ na None, aby zasymulować brakujące wartości:

Kod źródłowy:

```
iris_df.loc[0:3, 'sepal length (cm)'] = None
print("Dane przed oczyszczaniem:")
print(iris_df)
print("Ilość brakujących wartości każdej z cech:")
print(iris_df.isnull().sum())
```

Wynik:

```
Dane przed oczyszczaniem:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  NaN               3.5                1.4               0.2
1                  NaN               3.0                1.4               0.2
2                  NaN               3.2                1.3               0.2
3                  NaN               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[150 rows x 4 columns]
Ilość brakujących wartości każdej z cech:
sepal length (cm)    4
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Następnie brakujące dane postanowiłem dodać jako średnią z wartości danego atrybutu.

Kod źródłowy:

```
# Uzupełnianie brakujących wartości poprzez średnią danego atrybutu
iris_df['sepal length (cm)'].fillna(iris_df['sepal length (cm)'].mean(), inplace=True)
print("Dane po oczyszczaniu:")
print(iris_df)
print("Ilość brakujących wartości każdej z cech:")
print(iris_df.isnull().sum())
```

Wynik:

```
Dane po oczyszczaniu:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0             5.871233               3.5                1.4               0.2
1             5.871233               3.0                1.4               0.2
2             5.871233               3.2                1.3               0.2
3             5.871233               3.1                1.5               0.2
4             5.000000               3.6                1.4               0.2
..                 ...               ...                ...               ...
145           6.700000               3.0                5.2               2.3
146           6.300000               2.5                5.0               1.9
147           6.500000               3.0                5.2               2.0
148           6.200000               3.4                5.4               2.3
149           5.900000               3.0                5.1               1.8

[150 rows x 4 columns]
Ilość brakujących wartości każdej z cech:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Alternatywnie sprawdziłem również metodę usuwania wektorów cech, gdzie znajdują się niekompletne dane. W tym celu raz jeszcze zmanipulowałem dane wprowadzając brakujące wartości i przystąpiłem do czyszczenia.

Kod źródłowy:

```
# Usuwanie wierszy z brakującymi wartościami
iris_df.loc[0:3, 'sepal width (cm)'] = None
print("Dane przed oczyszczaniem:")
print(iris_df)
print("Ilość brakujących wartości każdej z cech:")
print(iris_df.isnull().sum())

iris_df.dropna(inplace=True)
print("Dane po oczyszczaniu:")
print(iris_df)
print("Ilość brakujących wartości każdej z cech:")
print(iris_df.isnull().sum())
```

Wynik:

```
Dane przed oczyszczaniem:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0             5.871233               NaN                1.4               0.2
1             5.871233               NaN                1.4               0.2
2             5.871233               NaN                1.3               0.2
3             5.871233               NaN                1.5               0.2
4             5.000000               3.6                1.4               0.2
..                 ...               ...                ...               ...
145           6.700000               3.0                5.2               2.3
146           6.300000               2.5                5.0               1.9
147           6.500000               3.0                5.2               2.0
148           6.200000               3.4                5.4               2.3
149           5.900000               3.0                5.1               1.8

[150 rows x 4 columns]
Ilość brakujących wartości każdej z cech:
sepal length (cm)    0
sepal width (cm)     4
petal length (cm)    0
petal width (cm)     0
dtype: int64
Dane po oczyszczaniu:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
4                  5.0               3.6                1.4               0.2
5                  5.4               3.9                1.7               0.4
6                  4.6               3.4                1.4               0.3
7                  5.0               3.4                1.5               0.2
8                  4.4               2.9                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[146 rows x 4 columns]
Ilość brakujących wartości każdej z cech:
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Ostatnią z metod czyszczenia danych było usunięcie duplikatów. W tym celu wczytałem raz jeszcze oryginalny zbiór danych Iris, połączyłem go z samym sobą, aby uzyskać duplikaty. A następnie posłużyłem się funkcją drop_duplicates() w celu pozbycia się ich.

Kod źródłowy:

```
# Usuwanie duplikatów
iris = datasets.load_iris()
iris_df = pd.DataFrame(iris.data, columns=iris.feature_names)
iris_df = pd.concat([iris_df, iris_df])
print("Dane zduplikowane:")
print(iris_df)
print("Liczba duplikatów przed usunięciem:")
print(iris_df.duplicated().sum())
iris_df.drop_duplicates(inplace=True)
print("Dane po usunieciu duplikatów:")
print(iris_df)
print("Liczba duplikatów po usunięciu:")
print(iris_df.duplicated().sum())
```

Wynik:

```
Dane zduplikowane:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[300 rows x 4 columns]
Liczba duplikatów przed usunięciem:
151
Dane po usunieciu duplikatów:
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

[149 rows x 4 columns]
Liczba duplikatów po usunięciu:
0
```