Klasteryzacja danych za pomocą sieci Kohonena

Aleksander Kłak, prowadzący Dr Marek Bazan

18 czerwca 2023

Spis treści

1 Cel ćwiczenia

Zadaniem jest zaimplementowanie algorytmu Kohonena w celu grupowania danych (klasteryzacji) oraz sprawdzenia jego działania na dwóch zestawach danych: zestawie danych Iris oraz zestawie danych giełdowych

2 Algorytm Kohonena

Algorytm Kohonena jest jednym z algorytmów samoorganizujących map i służy do grupowania danych na podstawie ich podobieństwa. W kontekście tego zadania należy zastosować algorytm Kohonena, aby znaleźć klastry (grupy) danych w zbiorach danych Iris oraz danych giełdowych.

3 Miary podobieństwa

Iloczyn skalarny: Miara podobieństwa, która oblicza iloczyn skalarny dwóch wektorów. Iloczyn skalarny między dwoma wektorami oblicza się jako sumę iloczynów odpowiadających sobie elementów obu wektorów. Iloczyn skalarny jest większy, gdy wektory są bardziej skierowane w podobnych kierunkach, a mniejszy, gdy wektory są bardziej skierowane w przeciwnych kierunkach.

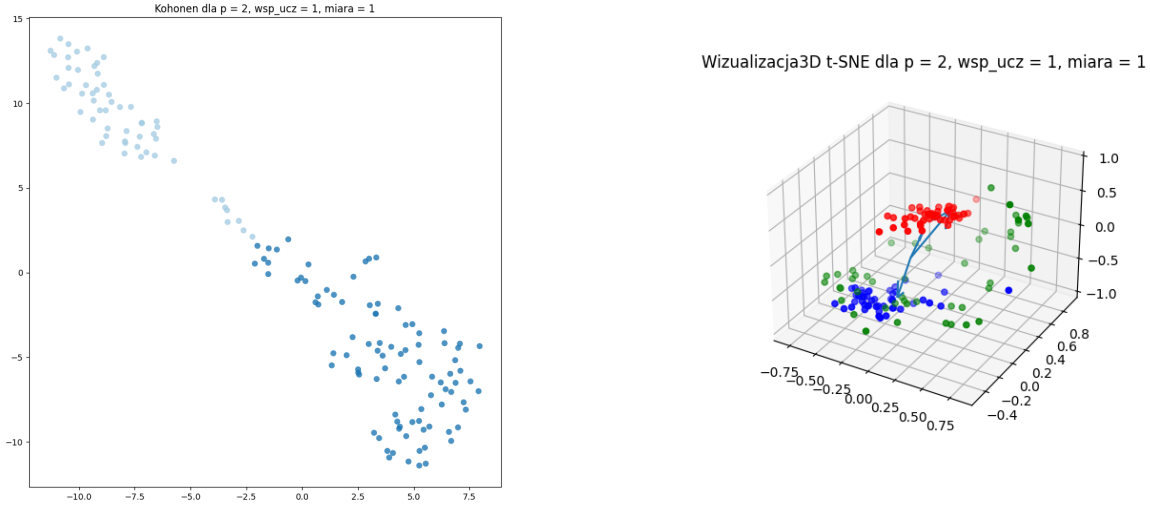
Odległość euklidesowa: Miara podobieństwa, która oblicza odległość pomiędzy dwoma punktami w przestrzeni. Odległość euklidesowa między dwoma wektorami oblicza się jako pierwiastek kwadratowy sumy kwadratów różnic między odpowiadającymi sobie elementami obu wektorów. Odległość euklidesowa jest niższa, gdy wektory są bardziej podobne i mają bliższe wartości elementów.

Odległość manhattańska oblicza sumę wartości bezwzględnych różnic między odpowiadającymi sobie elementami dwóch wektorów. Ta miara podobieństwa jest często używana w przypadkach, gdy istotne jest porównanie względnych różnic między wartościami elementów, a nie ich rzeczywiste wartości.

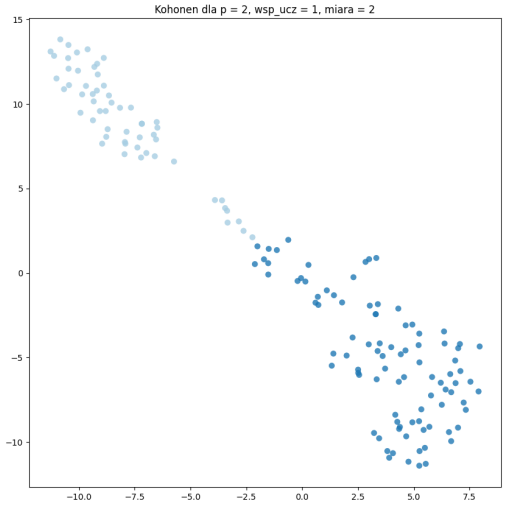
4 Wyniki

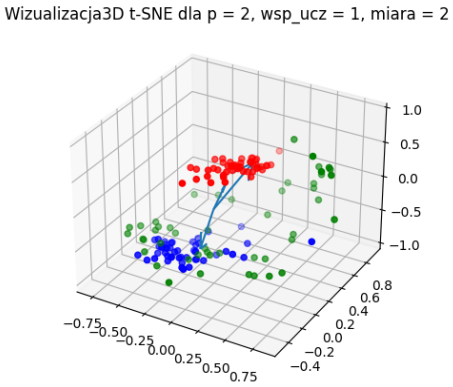
4.1 Dane Iris

4.1.1 Liczba klas: 2

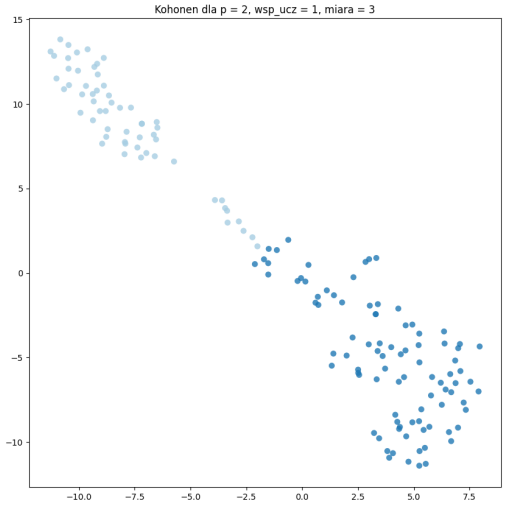


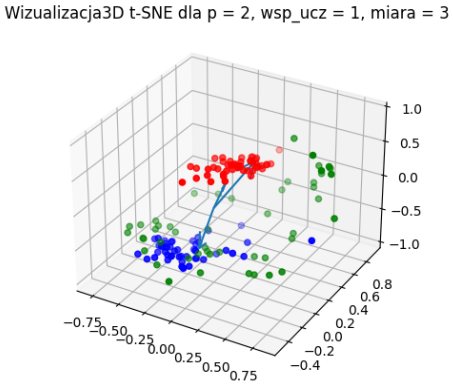
Rysunek 1: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa iloczyn skalarny



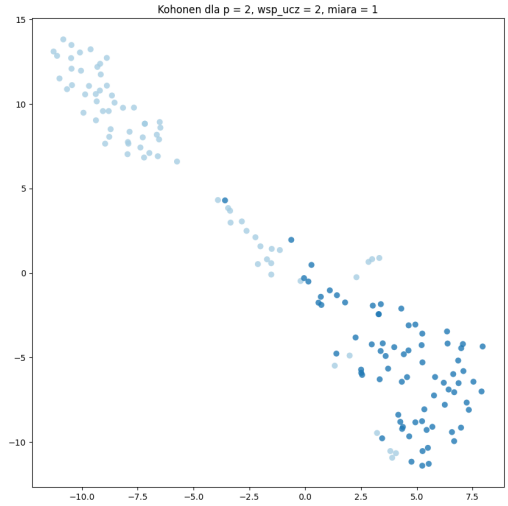


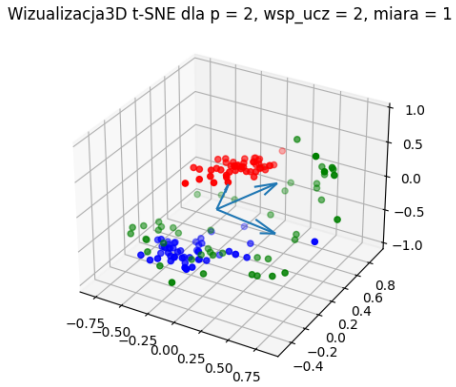
Rysunek 2: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa odległość euklidesowa



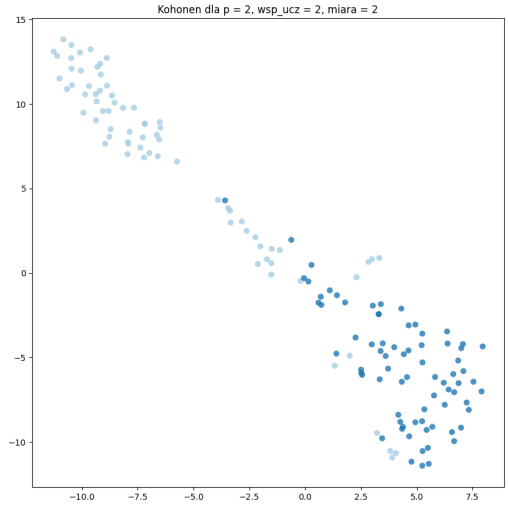


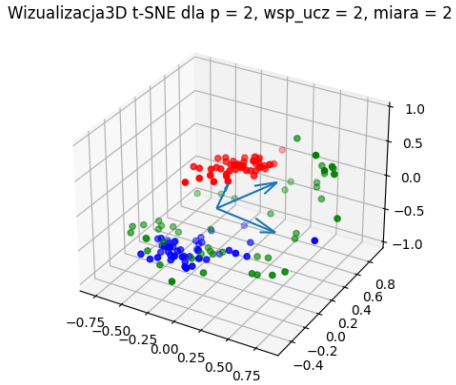
Rysunek 3: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa odległość manhattańska



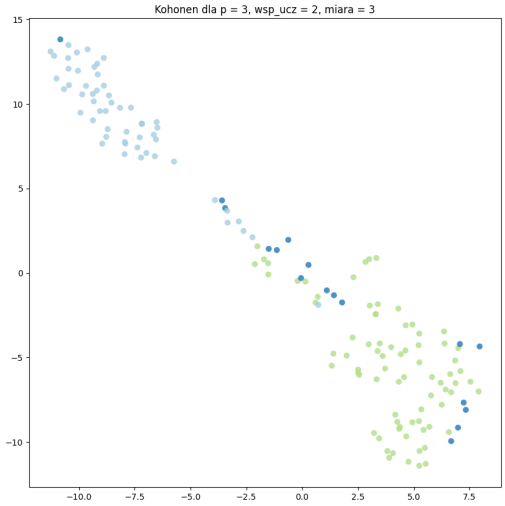


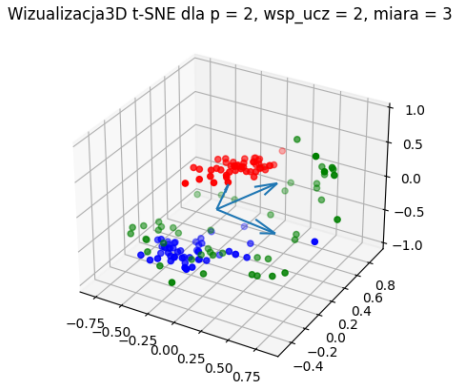
Rysunek 4: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa iloczyn skalarny



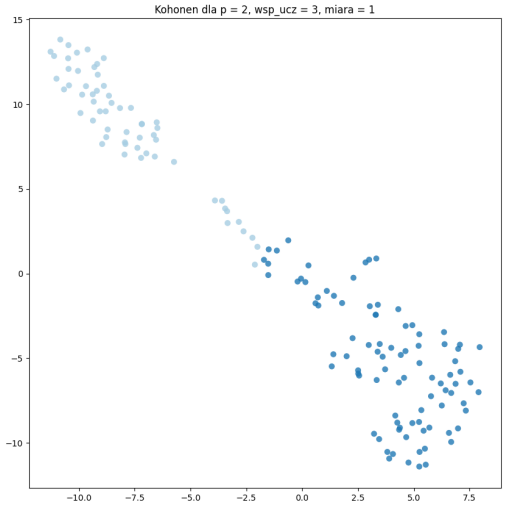


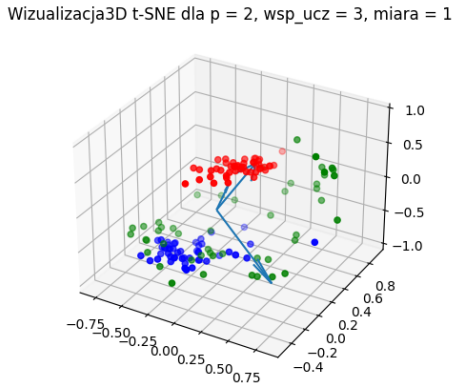
Rysunek 5: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa odległość euklidesowa



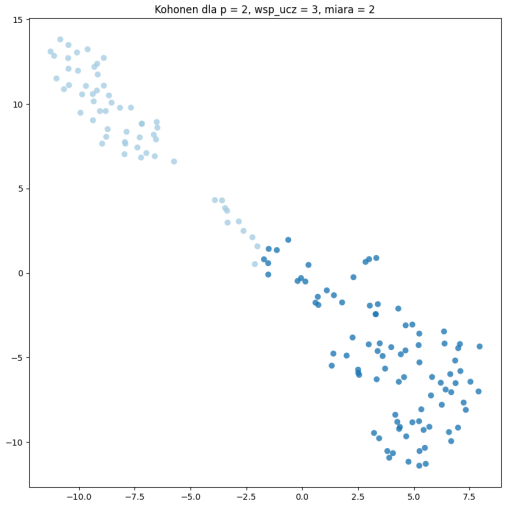


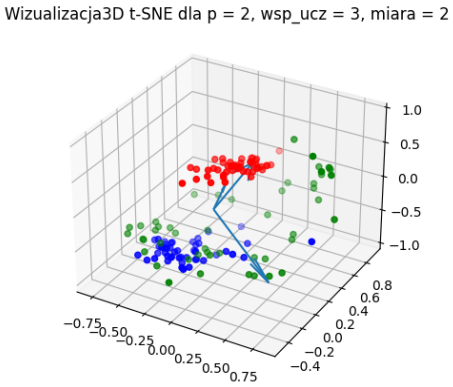
Rysunek 6: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa odległość manhattańska



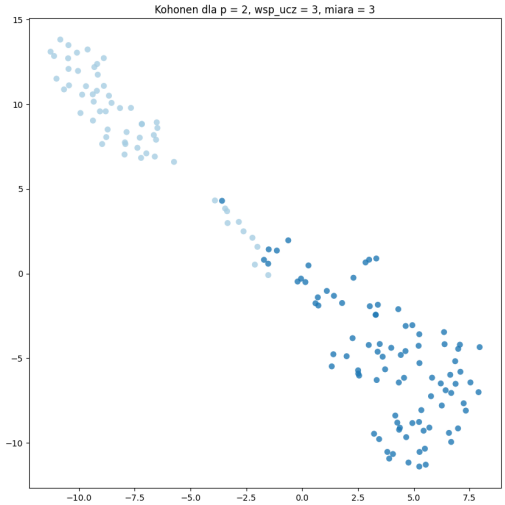


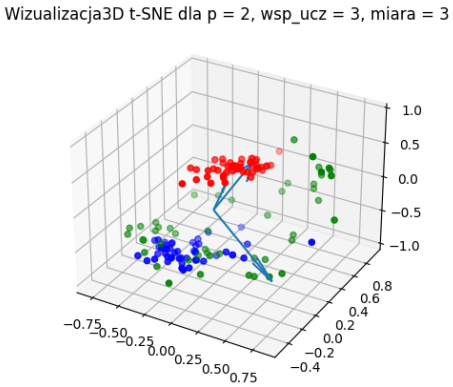
Rysunek 7: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa iloczyn skalarny





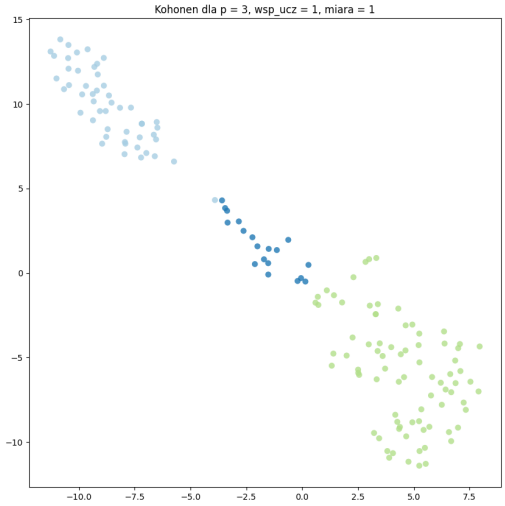
Rysunek 8: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa odległość euklidesowa

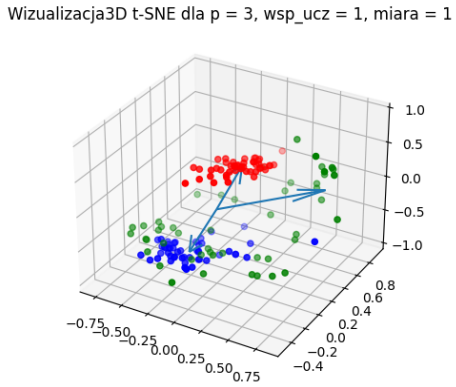




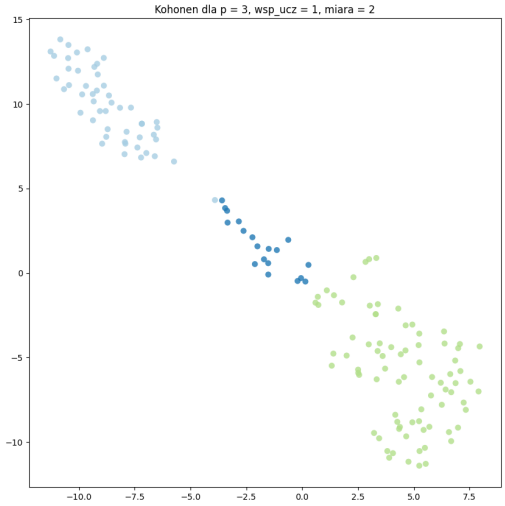
Rysunek 9: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa odległość manhattańska

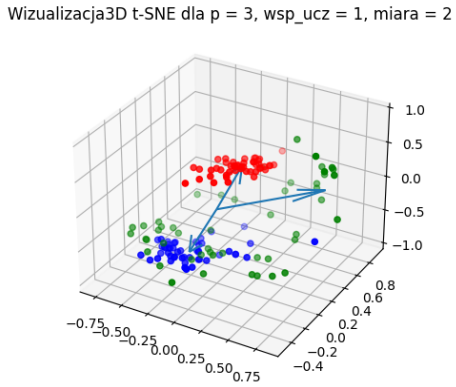
4.1.2 Liczba klas: 3



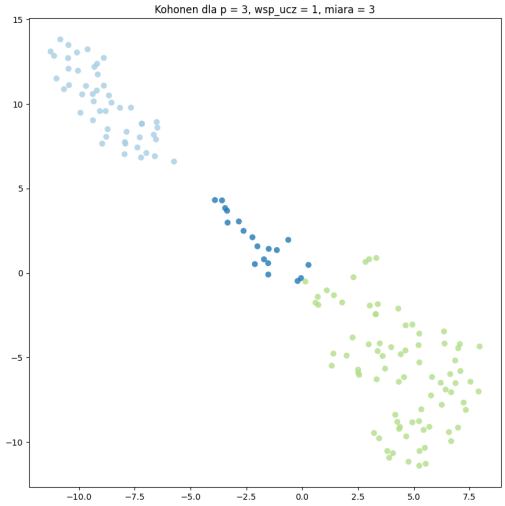


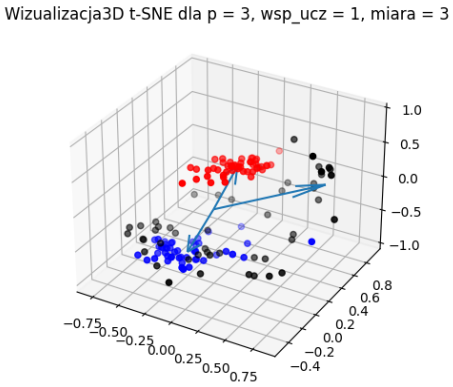
Rysunek 10: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa iloczyn skalarny



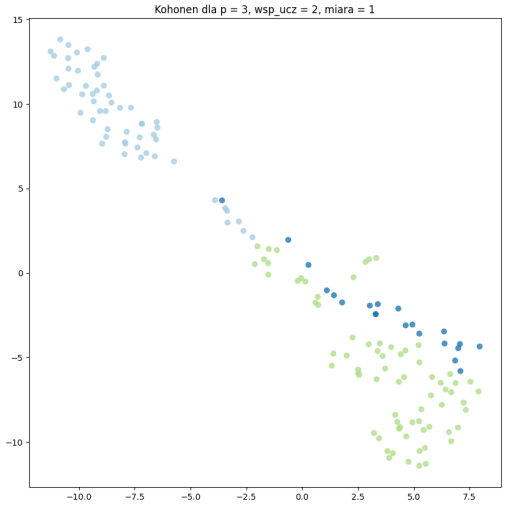


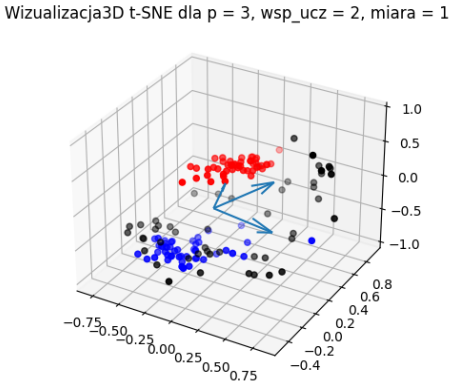
Rysunek 11: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa odległość euklidesowa



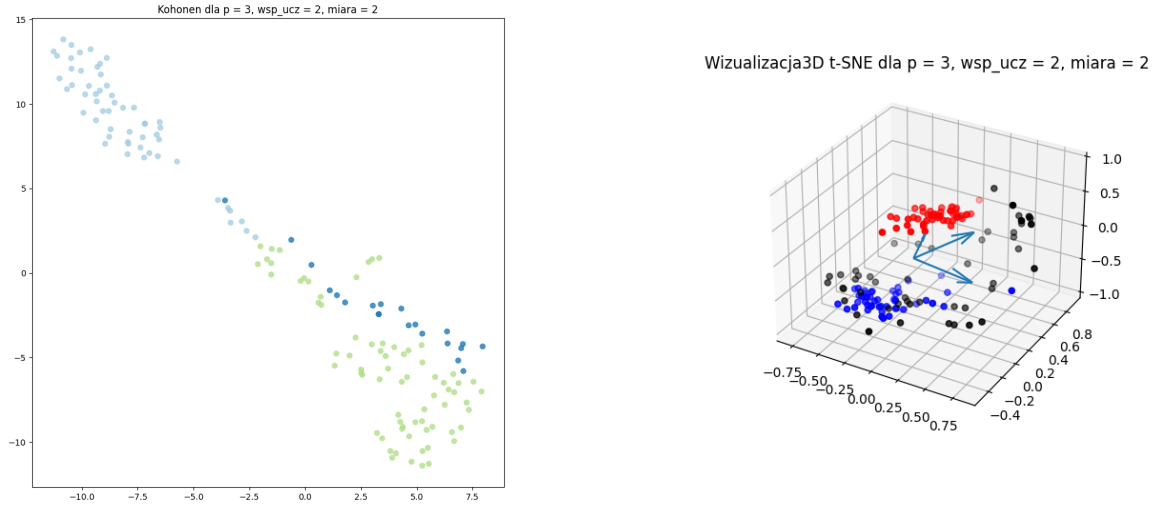


Rysunek 12: Współczynnik uczenia liniowo malejący (1), metoda miary podobieństwa odległość manhattańska

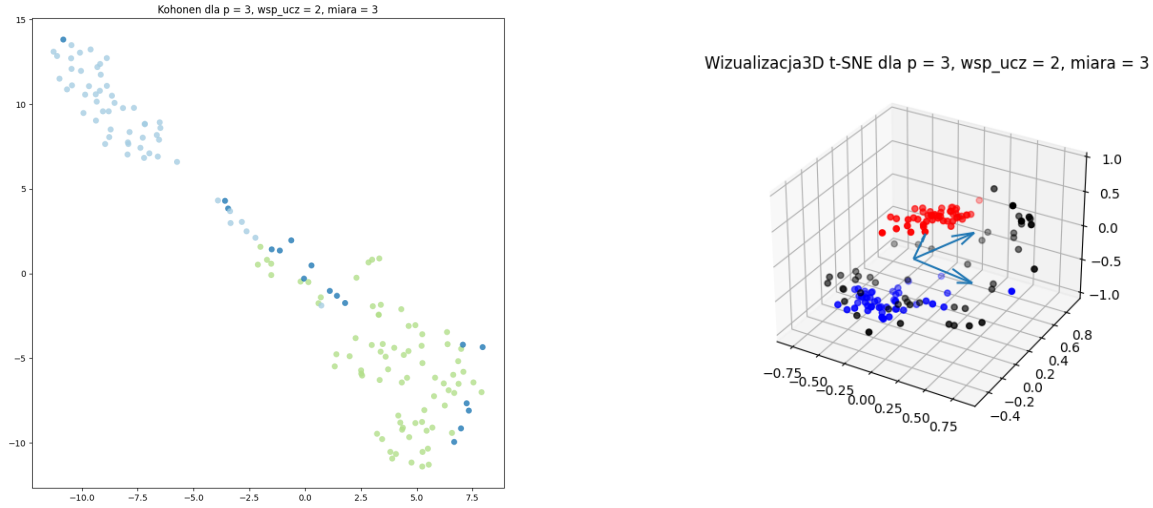




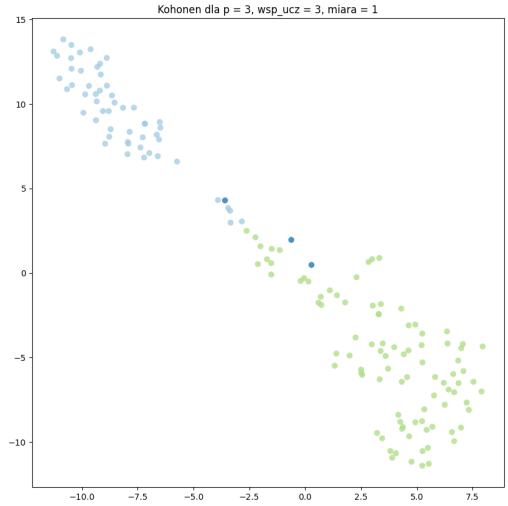
Rysunek 13: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa iloczyn skalarny

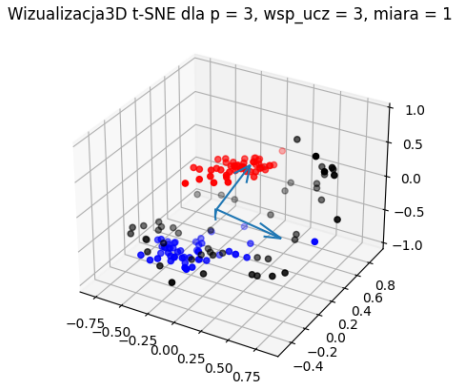


Rysunek 14: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa odległość euklidesowa

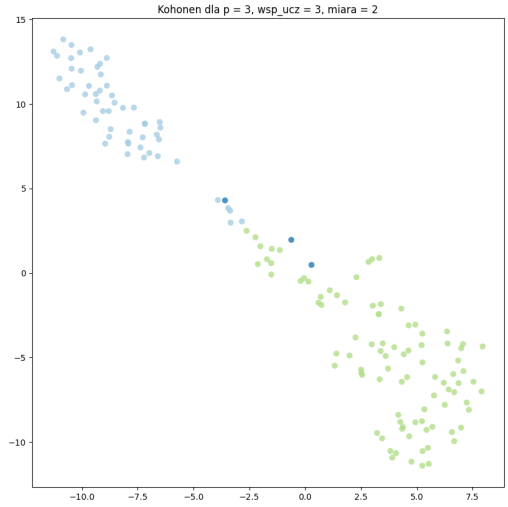


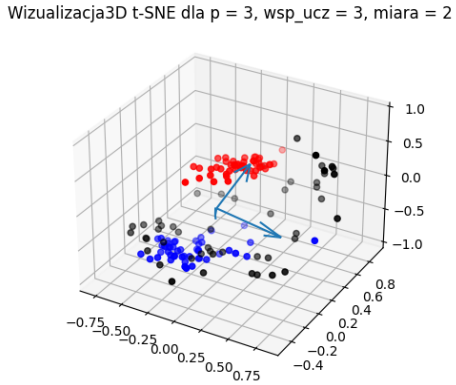
Rysunek 15: Współczynnik uczenia wykładniczo malejący (2), metoda miary podobieństwa odległość manhattańska



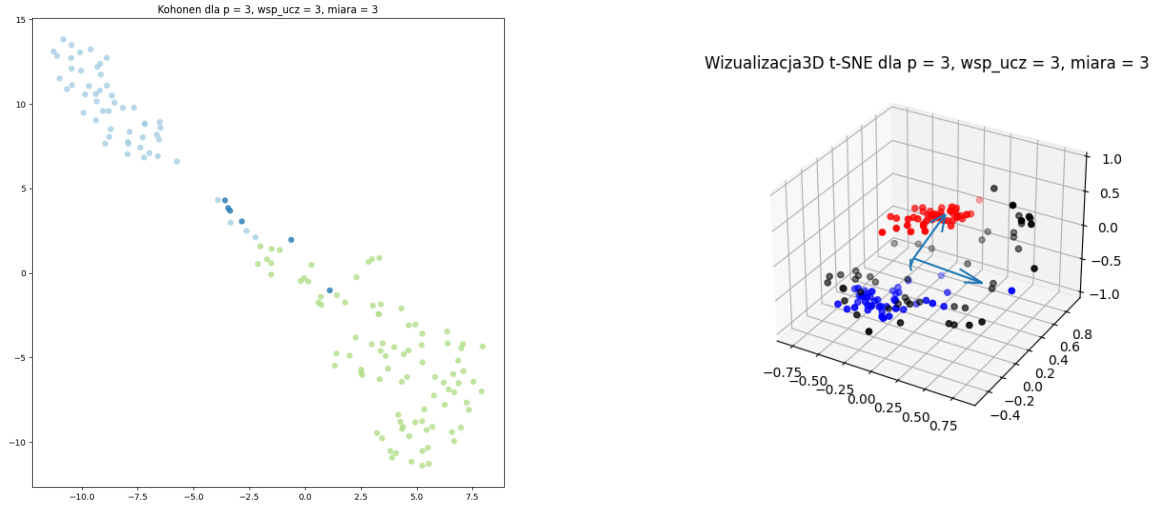


Rysunek 16: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa iloczyn skalarny





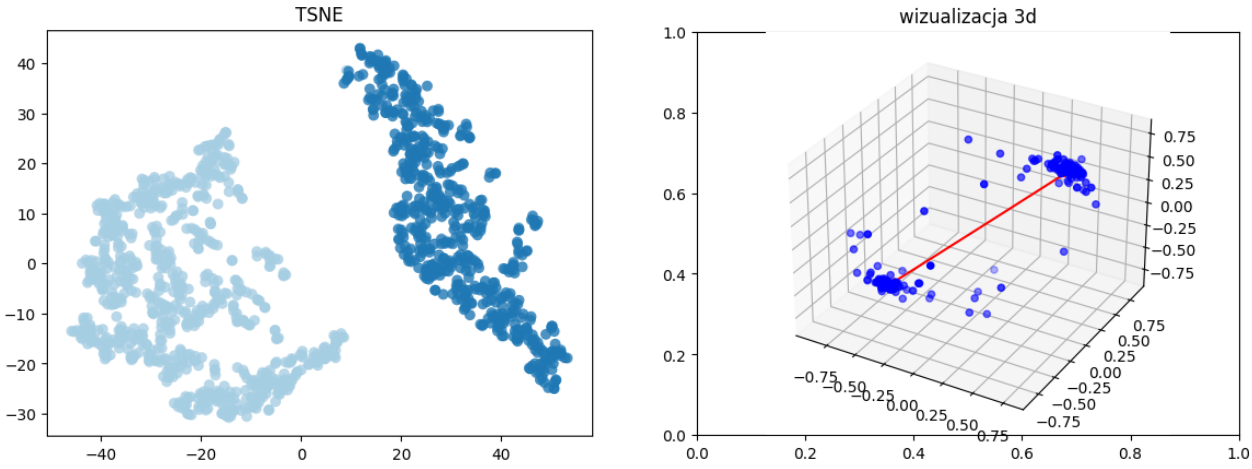
Rysunek 17: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa odległość euklidesowa



Rysunek 18: Współczynnik uczenia malejący hiperbolicznie (3), metoda miary podobieństwa odległość manhattańska

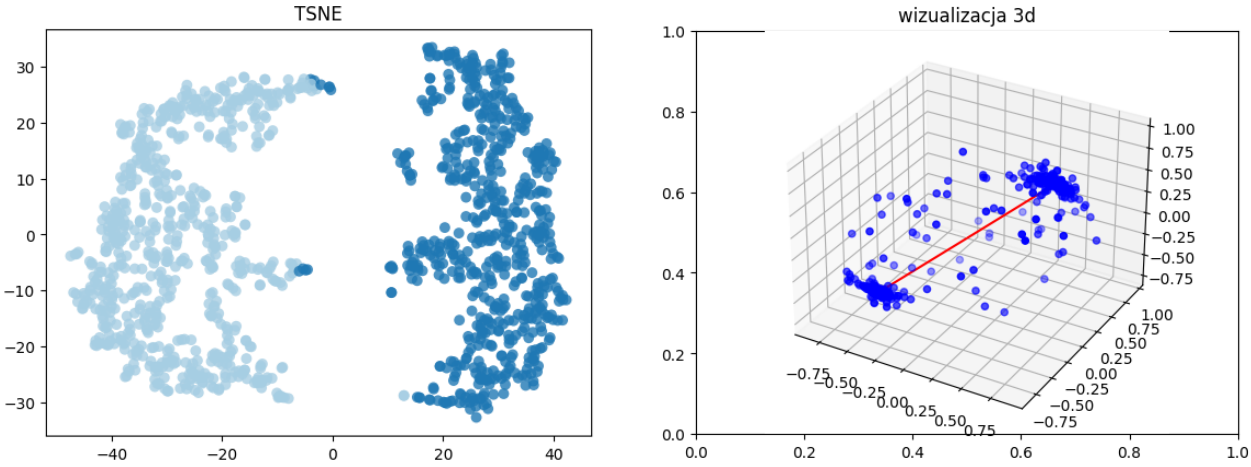
4.2 Dane giełdowe dla trzech spółek

4.2.1 Spółka AOS



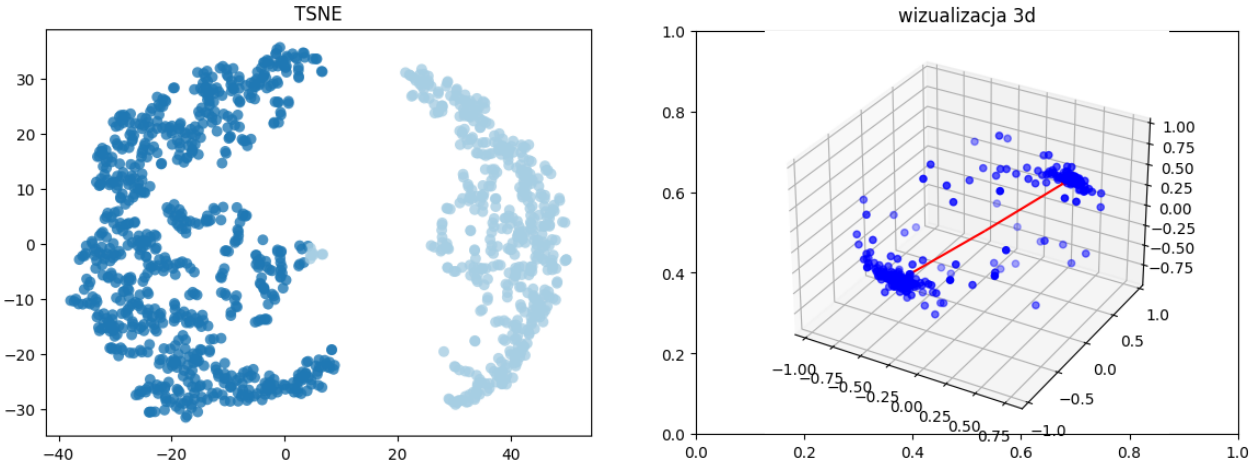
Rysunek 19: t-SNE, wizualizacja 3D, liniowo malejący, metoda manhattańska

4.2.2 Spółka AMP



Rysunek 20: t-SNE, wizualizacja 3D, liniowo malejący, metoda manhattańska

4.2.3 Spółka APD



Rysunek 21: t-SNE, wizualizacja 3D, liniowo malejący, metoda manhattańska

5 Wnioski

Dla zbioru irysów nie zaobserwowano różnicy między metodami iloczynu skalarnego i odległości euklideoswej. Zmiana współczynnika uczenia miała widoczny wpływ na umiejscowienie centrów danych.

6 Kod

6.1 Sposoby mierzenia odległości między danymi a reprezentantami

Listing 1: Metody mierzenia odlgegłości

```
def calc_scalar_product(X, T_value, num_repres, repres):
    measurements = [np.dot(rep, X[T_value % len(X)]) for rep in repres[:
        num_repres]]
    max_index = np.argmax(measurements)
    return max_index

def calc_euclidean_distance(X, T_value, num_repres, repres):
    measurements = [np.linalg.norm(rep - X[T_value % len(X)]) for rep in repres
        [:num_repres]]
    min_index = np.argmin(measurements)
    return min_index

def calc_absolute_difference(X, T_value, num_repres, repres):
    measurements = []

    for rep in repr[:num_repres]:
        absolute_diff = np.abs(rep - X[T_value % len(X)])
        measurements.append(np.sqrt(np.sum(absolute_diff)))

    min_index = np.argmin(measurements)
    return min_index

measure_methods = [calc_scalar_product, calc_euclidean_distance,
    calc_absolute_difference]
```

6.2 Algorytm Kohonena

Listing 2: Kohonen

```
def normalize_data(X):
    data_mean = np.mean(X, axis=0)
    X_normalized = (X - data_mean) / np.linalg.norm(X - data_mean, axis=1, keepdims=True)
    return X_normalized

def initialize_vectors(X, num_vectors):
    random_generator = np.random.RandomState(0)
    vectors = random_generator.normal(loc=0.0, scale=0.01, size=(num_vectors, len(X[0])))
    vectors /= np.linalg.norm(vectors, axis=1, keepdims=True)
    return vectors

def update_learning_rate(learning_rate_option, initial_learning_rate, C1, C2,
    current_iteration, total_iterations):
    if learning_rate_option == 1:
        return initial_learning_rate * (total_iterations - current_iteration) /
            total_iterations
    elif learning_rate_option == 2:
        return initial_learning_rate * math.exp(-C1 * current_iteration)
    elif learning_rate_option == 3:
        return C1 / (C2 + current_iteration)
    else:
        raise ValueError("Invalid learning_rate_option")

def Kohonen(X, T, initial_learning_rate, num_vectors, learning_rate_option, measure_option,
    C1, C2, normalize):
    """
    X - dane
    T - liczba iteracji
    init_lr - wspolczynnik uczenia
    num_vecs - liczba reprezentant w
    lr_option - 1 - malejacy, 2 - wykladniczy, 3 - hiperboliczne zmniejszanie
    measure_opt - 1 - skalar, 2 - euklidesowa, 3 - manhattanska
    C1 - parametr dla lr_option = 2
    C2 - parametr dla lr_option = 3
    norm - czy normalizowac dane
    """
    if normalize:
        X = normalize_data(X)

    vectors = initialize_vectors(X, num_vectors)

    measurements_table = np.zeros(len(X))

    for iteration in range(T):
        selected_index = measure_option(X, iteration, num_vectors, vectors)
        measurements_table[iteration % len(X)] = selected_index

        vectors[selected_index] += initial_learning_rate * (X[iteration % len(X)] - vectors[
            selected_index])
        vectors[selected_index] /= np.linalg.norm(vectors[selected_index])

        initial_learning_rate = update_learning_rate(learning_rate_option,
            initial_learning_rate, C1, C2, iteration, T)

    return vectors, measurements_table, X
```

6.3 Skrypt

Listing 3: Skrypt

```
# Definicja parametrów
num_iterations = 10000, learning_rate_initial = 0.1, num_representatives = 2
learning_rate_type = 1, measure_method = measure_methods[1]
param_C1 = 0.3, param_C2 = 0.4
normalize_data = True
plot_title = 'p=2 IRIS LR=1, MO=2'

# Wczytanie danych
dataset = datasets.load_iris()
data = dataset.data

# Obliczenie Kohonena
kohonen_result, predictions, transformed_data = Kohonen(
    data,
    num_iterations,
    learning_rate_initial,
    num_representatives,
    learning_rate_type,
    measure_method,
    param_C1,
    param_C2,
    normalize_data
)

# Wykresy
predictions_np = np.array(predictions)
plt.figure()
plt.plot(predictions_np, 'o',  markersize=2)
plt.vlines([50,100],0,3, colors='r', linestyles='dashed')
plt.title('Kohonen -"'.format(plot_title))
plt.xlabel('Dane', fontsize=15)
plt.ylabel('Przypisanie ', fontsize=15)
plt.show()

# Wykresy t-SNE
tsne_result = TSNE(random_state=1).fit_transform(transformed_data)
plt.figure()
plt.scatter(tsne_result[:,0], tsne_result[:,1], lw=0, s=50, alpha=0.8, edgecolors='black',
            c=np.array(sns.color_palette("Paired"))[predictions_np.astype(np.int)])
plt.title('TSNE -"'.format(plot_title))
plt.show()

# Wykresy 3D
center = [0,0,0]
fig = plt.figure()
plt.title('Wizualizacja 3D -"'.format(plot_title))
ax = fig.add_subplot(111,projection='3d')
u1, v1, w1 = zip(center, center)
u2, v2, w2 = zip(kohonen_result[0, 0:3], kohonen_result[1, 0:3])
ax.quiver(u1, v1, w1, u2, v2, w2, arrow_length_ratio=0.1, colors='black')
ax.scatter(transformed_data[0:49,0], transformed_data[0:49,1], transformed_data[0:49,2], c='r
    ')
ax.scatter(transformed_data[50:99,0], transformed_data[50:99,1], transformed_data[50:99,2], c
    ='g')
ax.scatter(transformed_data[99:149,0], transformed_data[99:149,1], transformed_data
    [99:149,2], c='b')
plt.show()
```