KMeans

`**KMeans**` to klasa implementująca algorytm k-średnich (k-means), który jest używany do grupowania danych na podstawie ich cech.

Metody

- `**__init__(self, liczba_klastrow, max_iter=100)**`: Konstruktor klasy, który przyjmuje liczbę klastrów (`**liczba_klastrow**`) do wygenerowania oraz maksymalną liczbę iteracji (`**max_iter**`), po której algorytm się zatrzyma, jeśli nie znajdzie lepszego podziału.

- `**inicjalizuj_centroidy(self, dane)**`: Metoda do inicjalizacji początkowych centroidów poprzez losowe wybranie punktów z danych.

- `**przydziel_klastry(self, dane)**`: Metoda obliczająca dystans każdego punktu danych do każdego centroidu i przypisująca punkt do klastra z najbliższym centroidem.

- `**aktualizuj_centroidy(self, dane, etykiety)**`: Metoda aktualizująca położenie centroidów na podstawie obecnych przypisań punktów do klastrów.

- `**dopasuj(self, dane)**`: Główna metoda, która wywołuje powyższe metody w celu wykonania klastrowania danych. Zwraca etykiety klastrowe dla każdego punktu danych.

SiecNeuronowaPNN

`**SiecNeuronowaPNN**` to klasa reprezentująca Probalistyczną Sieć Neuronową (PNN), która klasyfikuje dane na podstawie nauczonego modelu.

Metody

- `**__init__(self, sigma=1.0)**`: Konstruktor klasy, który przyjmuje parametr `sigma` używany w funkcji Gaussa (jądra Gaussowskiego).

- `**dopasuj(self, dane, etykiety)**`: Metoda służąca do trenowania sieci. Przyjmuje dane oraz ich etykiety i organizuje je według klas.

- `**funkcja_gaussa(self, odleglosc)**`: Metoda obliczająca wartość funkcji Gaussa dla danej odległości, która jest wykorzystywana do obliczenia prawdopodobieństwa przynależności punktu do klasy.

- `**przewiduj_prawdopodobienstwa(self, x)**`: Metoda obliczająca prawdopodobieństwo przynależności nowego punktu `**x**` do każdej z klas. Zwraca słownik z prawdopodobieństwami dla każdej klasy.

Wykorzystanie

- Najpierw generujemy losowe dane dla trzech różnych kształtów (trójkątów, kółek, krzyżyków) i umieszczamy je na płaszczyźnie 2D.

- Tworzymy instancję klasy **`KMeans`** z trzema klastrami i dopasowujemy dane do modelu k-średnich, co pozwala na ich podział na grupy.

- Następnie tworzymy instancję klasy **`SiecNeuronowaPNN`** i trenujemy ją na podstawie danych i etykiet wygenerowanych przez algorytm k-średnich.

- Po wytrenowaniu sieci PNN, testujemy ją na nowym punkcie reprezentującym trójkąt, obliczając prawdopodobieństwo przynależności tego trójkąta do każdego z klastrów.

Wyniki

- Wyniki są wypisywane jako słownik prawdopodobieństw, gdzie klucze odpowiadają numerom klastrów, a wartości to obliczone prawdopodobieństwa przynależności nowego punktu do tych klastrów.

Kod ten służy jako prosta, ale pełna implementacja algorytmu k-średnich i PNN, zapewniając podstawowe narzędzia do klastrow