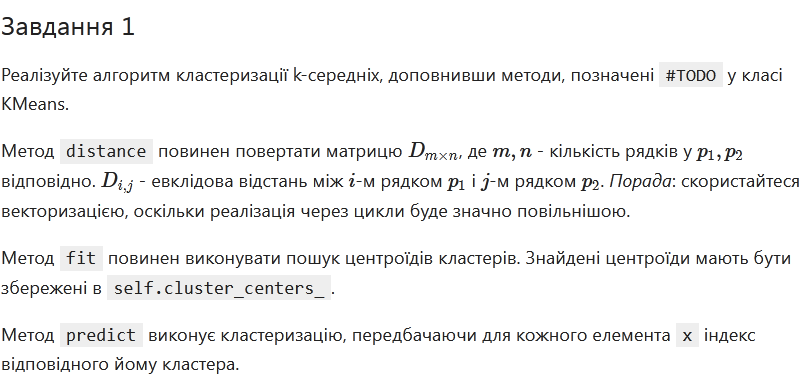
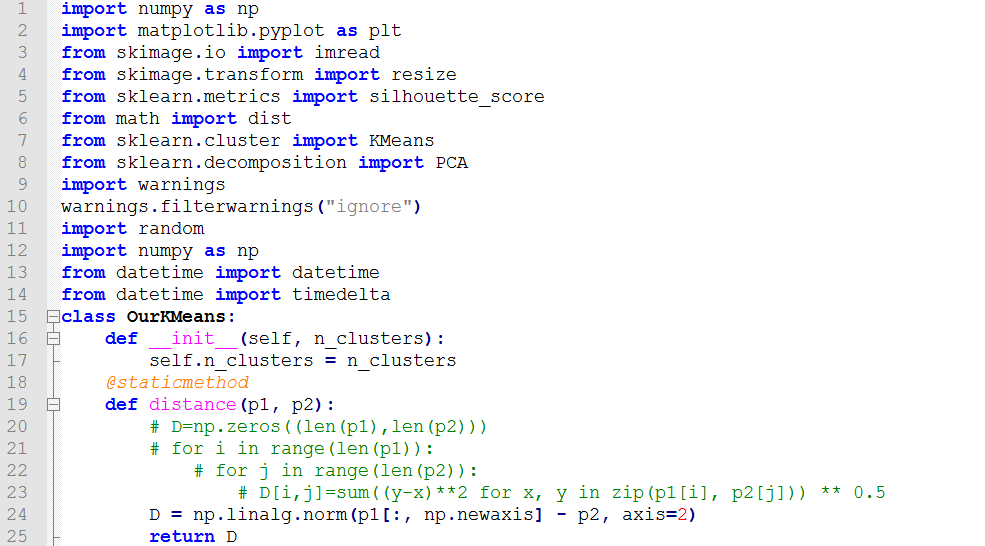
**Лабораторна робота 4 - Навчання без учителя.**

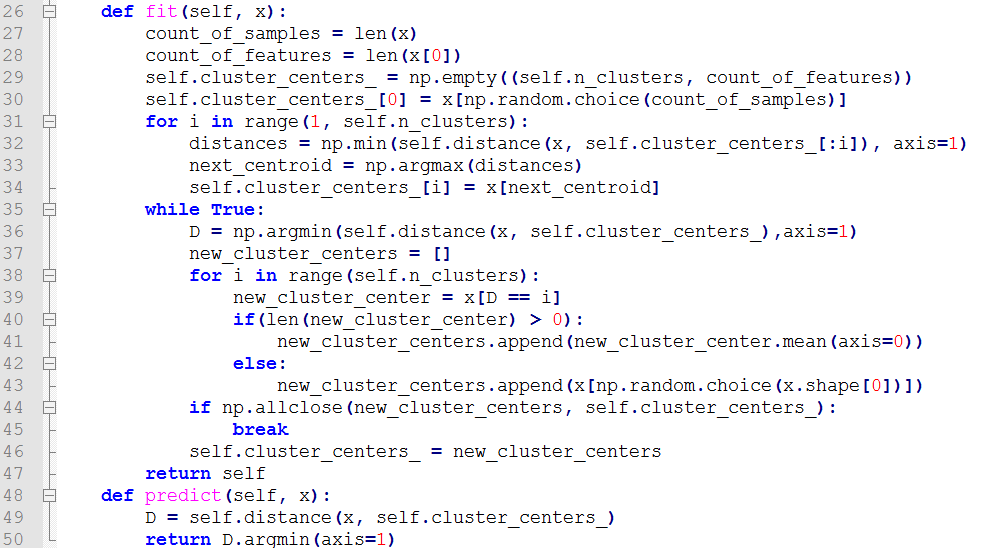
У цій лабораторній роботі Ви познайомитеся з основними задачами, які розв'язують алгоритми навчання без учителя.



Виберіть будь-яке зображення (тільки не дуже велике, 64х64 підійде ідеально). За допомогою алгоритму k-середніх підберіть оптимальну кількість кластерів для кластеризації пікселів зображення, максимізуючи silhouette-score. Візуалізуйте кластеризацію з найкращим k.

**Код класу OurKMeans:**





**Опис проведених досліджень**

KMeans, кластеризація методом К-середніх – популярний метод кластеризації, тобто впорядкування множини об'єктів у порівняно однорідні групи.

Реалізація алгоритму KMeans складається з таких основних частин:

1. Обчислення Евклідової відстані між двома точками за формулою:



2. Ініціалізація центрів кластерів. Спочатку використовувалось таке рішення:



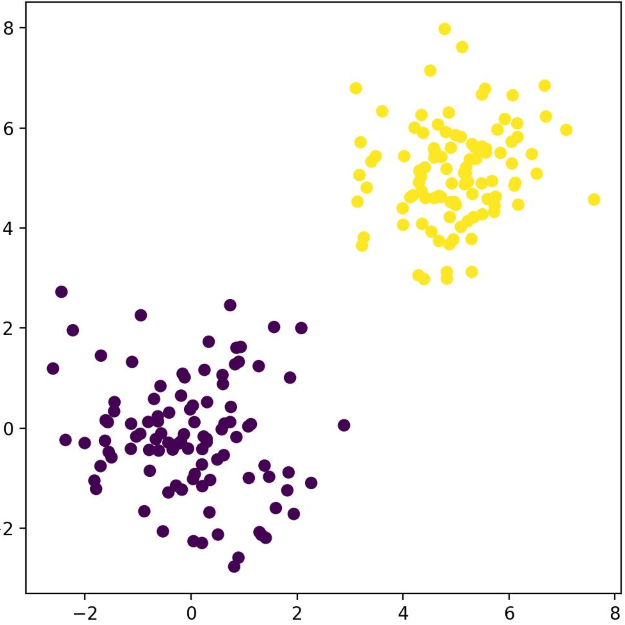
Воно коректно працює лише для набору випадкових даних, а при обробці зображення виникає помилка «Error: Number of labels is 1. Valid values are 2 to n\_samples – 1». Тому ми використовували інший алгоритм, K-means++. Він є більш розумним способом ініціалізації центроїдів. Замість того, щоб обирати їх випадковим чином, він випадково визначає лише перший, а наступні – на основі максимальної відстані від вже визначених. Для зменшення ймовірності виникнення помилки у коді також було додано обробку порожніх кластерів.

3. Призначення кожної точки набору даних *х* найближчому центру кластера, що виконується шляхом обчислення мінімальної відстані для кожної точки.

4. Оновлення центрів: після присвоєння точок кластерам ми обчислюємо нові центри кластерів як середні значення точок, призначених кожному кластеру.

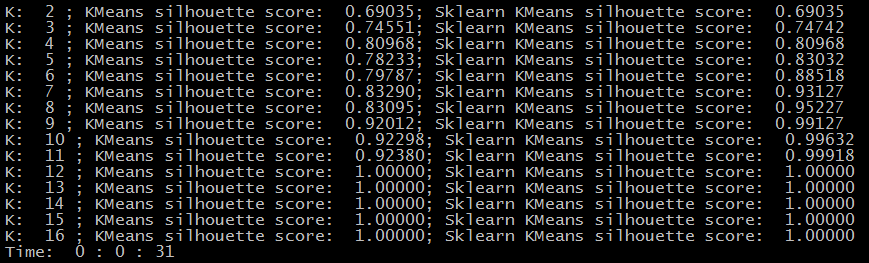
5. Перевірка збіжності: обчислення виконуються, доки центри не зміняться.

При перевірці роботи методів класу на випадкових даних при *K* = 2 було отримано такий графічний результат:

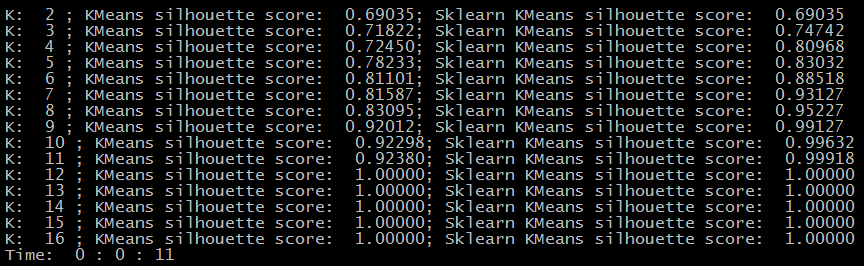


Для тестування класу використовувалось зображення розміру 64 на 64 у форматах bmp та jpg. У процесі виконання даної лабораторної роботи ми порівнювали silhouette-score, отримане нашим рішенням, із результатами роботи аналогічного інструмента KMeans із бібліотеки Sklearn. Також було реалізовано дві версії метода обчислення відстані: на основі циклів із оптимізацією обчислення квадратного кореня шляхом піднесення підкореневого виразу у степінь 0.5 замість використання команди sqrt, та команди np.linalg.norm.

Результати роботи програми із обчисленням відстані за допомогою циклів:

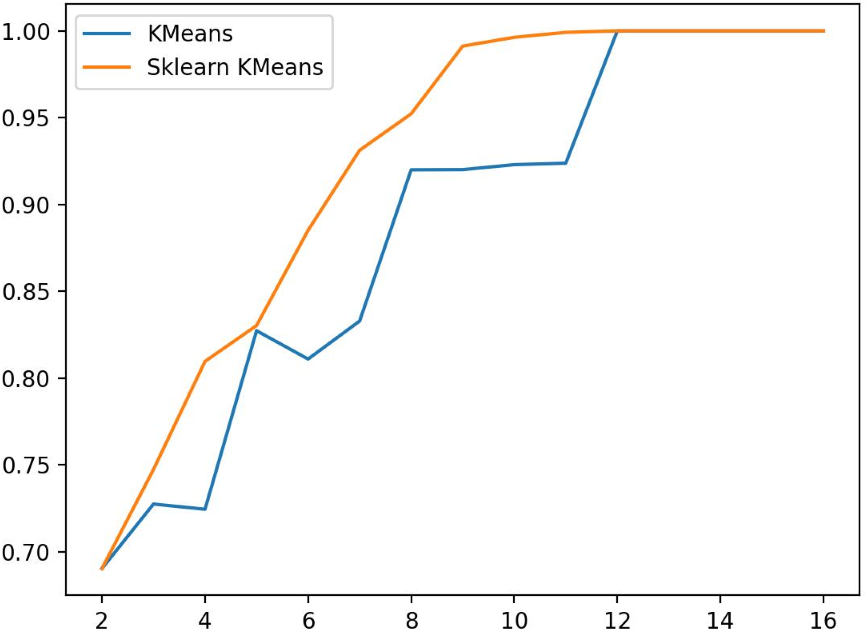


Результати роботи програми із обчисленням відстані без циклів:

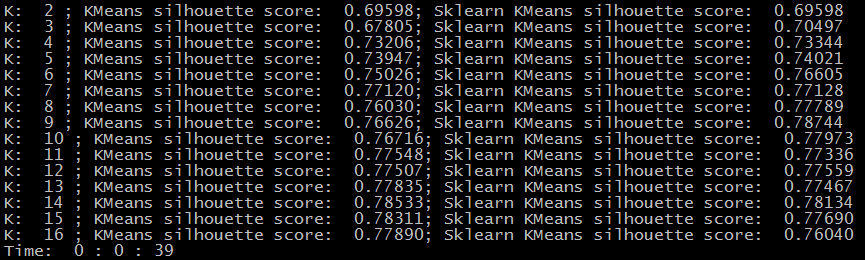


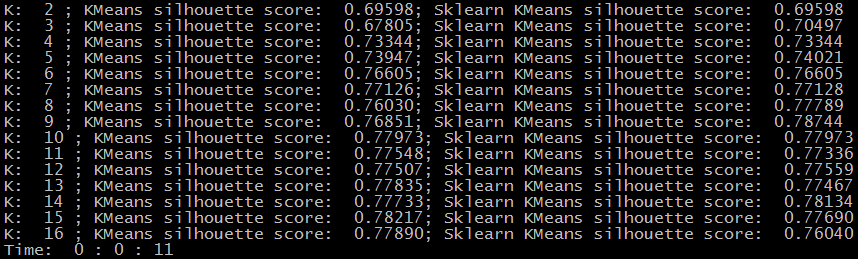
Як бачимо, версія із циклами працює майже в три рази повільніше, ніж відповідна функція бібліотеки numpy. Різні способи обчислення відстані дають різні результати при деяких значеннях *К*, а функція KMeans бібліотеки Sklearn показує кращий silhouette-score при всіх K окрім 2, що помітно на графіках:

|  |  |
| --- | --- |
| KMeans | Sklearn KMeans |
|  |  |



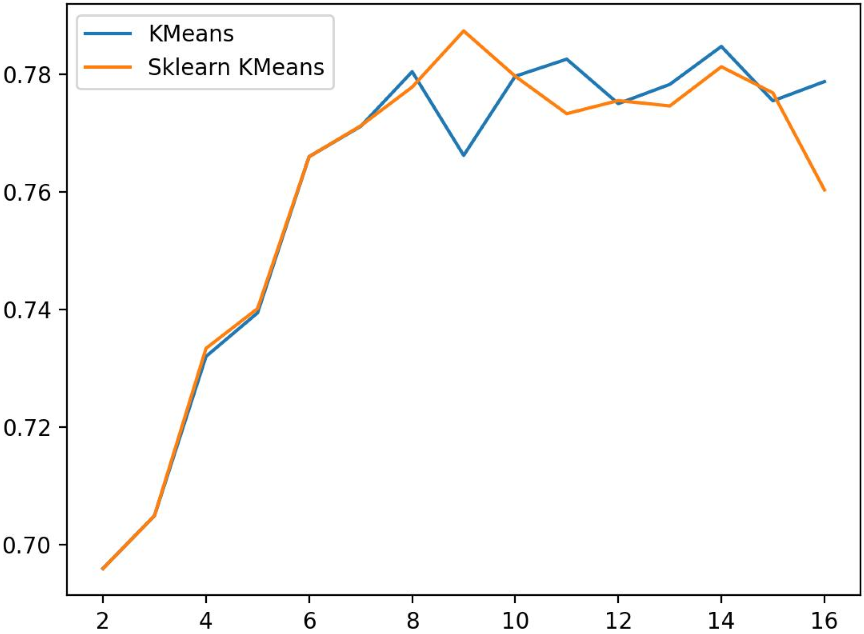
Результати експериментів із тим самим зображенням, але у форматі jpg:

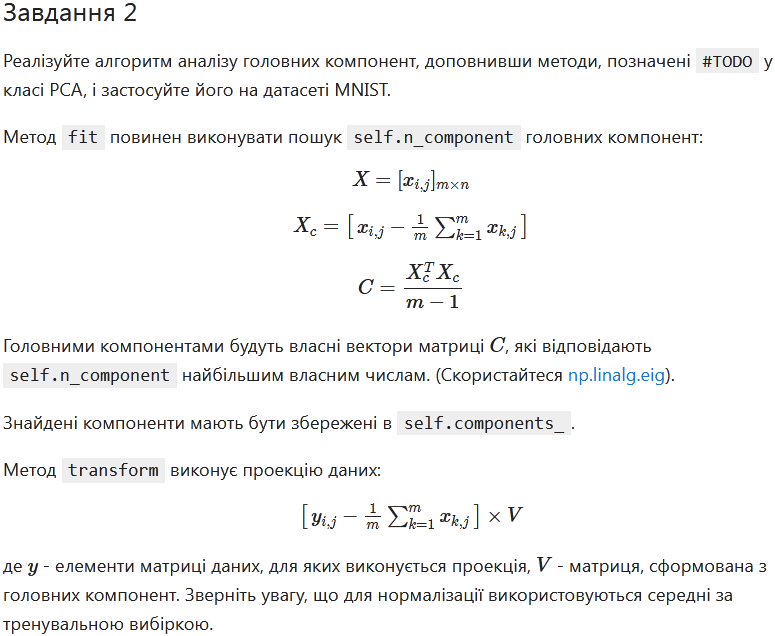




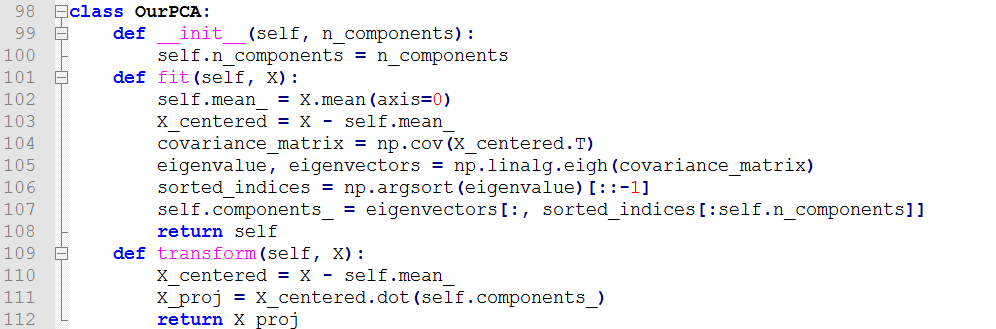
Як бачимо, загальний рівень silhouette-score для jpg нижчий за bmp. Також слід відмітити різні значення цього коефіцієнта у нашій реалізації при обчисленні відстані з циклами та без них. Співвідношення швидкодії приблизно дорівнює bmp.

На цьому графіку можна помітити, що у деяких випадках наша реалізація KMeans дає кращий silhouette-score:





**Код класу OurPCA:**



**Опис проведених досліджень**

PCA (principal component analysis), метод головних компонент – один із основних способів зменшення розмірності даних із втратою найменшої кількості інформації. Його реалізація складається з таких основних етапів:

1. центрування даних (віднімання середнього значення кожної ознаки);

2. обчислення коваріаційної матриці;

3. розкладання коваріаційної матриці на власні значення (eigenvalues);

4. сортування векторів власних значень (eigenvectors) за власними значеннями (eigenvalues) в порядку спадання;

5. Проекція даних на основні компоненти (метод *transform*).

У рамках даної частини лабораторної роботи ми порівняли нашу реалізацію та подібний інструмент з бібліотеки Sklearn, проаналізувавши датасет MNIST:

|  |  |
| --- | --- |
| PCA | Sklearn PCA |
|  |  |

Як бачимо, графічний результат роботи нашої версії PCA є дзеркальним відображенням Sklearn PCA.