Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

ЗВІТ

про виконання лабораторної роботи № 3
з дисципліни «Інтелектуальний аналіз даних»

Виконала:
Студентка III курсу
Групи КА-76
Оркуша А. Д.

Перевірила:
Недашківська Н. І.

Київ – 2020

Варіант 11

Алгоритм Spectral clustering.

Метрики якості: Estimated number of clusters, Adjusted Rand Index, Silhouette Coefficient, Davies-Bouldin index.

Чи є розбиття стабільним після вилучення окремих об’єктів?
Початкові дані:

(а) sklearn.datasets.load_digits
(б) sklearn.datasets.make_moons

Хід виконання роботи:

1. Представити початкові дані графічно.
2. Побудувати модель кластеризації згідно з варіантом.
3. Виконати кластеризацію даних на основі моделі.
4. Представити розбиття на кластери графічно, наприклад, різними кольорами.
5. Розрахувати додаткові результати кластеризації згідно з варіантом.
6. Побудувати декілька альтернативних моделей:
    - шляхом зміни значень параметрів основної моделі,
    - використати різні функції відстані,
    - задати різні значення кількості кластерів, в алгоритмах де кількість кластерів - параметр.

7. Для кожної альтернативної моделі розрахувати метрики якості кластеризації, що реалізовані в класі metrics, згідно з варіантом:

    - Estimated Number of Clusters.
    - Adjusted Rand Index.
    - Silhouette Coefficient
    - Davies-Bouldin index.

8. Виконати аналіз результатів кластеризації одним з неформальних мето-

дів згідно з варіантом:

Чи є розбиття стабільним після видалення окремих об'єктів?

9. Зробити висновки про якість роботи моделей на досліджених даних. Дослідити різні значення параметрів основної моделі, різні функції відстані та різну кількість кластерів в алгоритмах, де кількість кластерів слугує параметром.

10. Оцінити результати кластеризації на основі метрик якості та на основі неформальних методів. Для кожного набору даних вибрати найкращу модель.

Виконання

Почнемо з опису спектральної кластеризації.

Спектральна кластеризація - техніка з коренями в теорії графів, в якій використано підхід що ідентифікує скупчення вершин в графі, аналізуючи ребра що їх поєднують. Метод доволі гнучкий і дозволяє також кластеризувати дані що не представлені у вигляді графів.

Спектральна кластеризація використовує підраховані власні значення та вектори(спектр) спеціальних матриць, побудованих на основі вхідних даних. Власні вектори - важлива частина лінійної алгебри, оскільки вони допомагають описати динаміку системи яку репрезентує матриця.

У версії цього алгоритму, що імплементована у бібліотеці sklearn використовується матриця подібності, яку можна підрахувати самостійно і передати як параметр, або обрати один з методів її підрахунку з модуля `sklearn.metrics.pairwise`. Також можна обрати метод підрахунку власних значень та векторів.

Алгоритм:

На вхід передається матриця подібності $S \in R^{n \times n}$ і кількість кластерів k

1. Конструюємо граф подібності з матриці подібності, нехай W його зважена матриця суміжності
2. Підраховуємо нормалізований лапласіан Lsym
3. Підраховуємо перші k власні вектори u1, . . . , uk матриці Lsym.

4. Формуємо матрицю U, стовпчики якої - підраховані власні вектори u1, . . . , uk
5. Формуємо матрицю T(nxk) нормуванням рядків матриці U
   $t_{ij} = u_{ij}/(\Sigma_k u_{ik}^2)^{1/2}$
6. Для i = 1, . . . , n нехай yi - вектор що відповідає і-му рядку матриці T
7. Кластеризуємо yi за допомогою алгоритму к-середніх в кластери C1,...,Ck
8. Формуємо вихідні дані: кластери A1, . . . , Ak де Ai = {j| yj ∈ Ci}.

Спектральна кластеризація - гнучкий підхід для знаходження кластерів коли дані не задовольняють вимогам інших поширених алгоритмів.

1. Представити початкові дані графічно

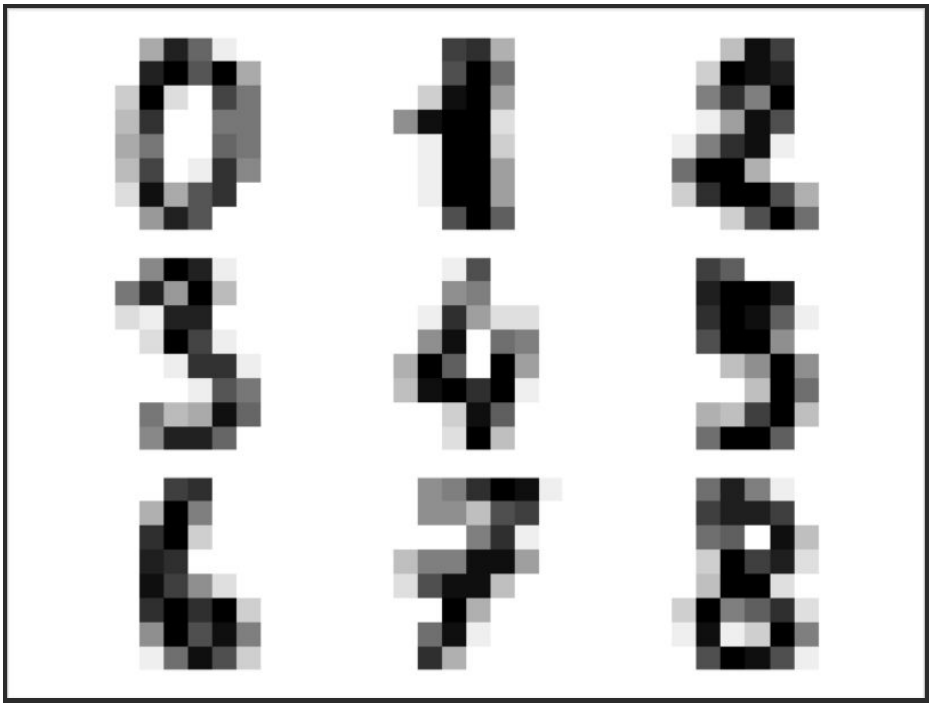
(a)load_digits

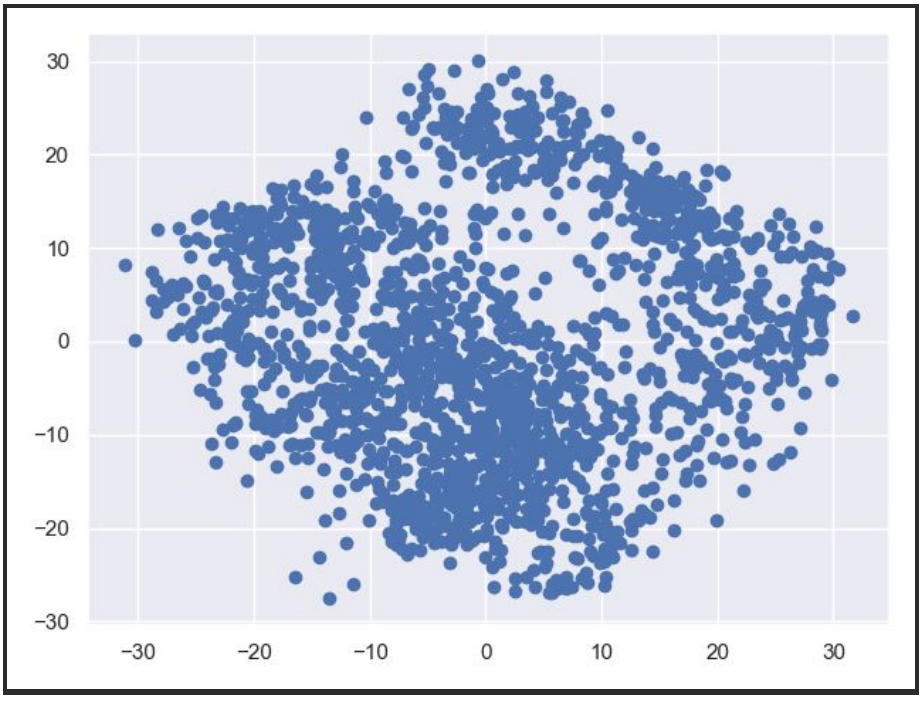
Цей датасет складається з набору n_samples зображень рукописних цифр (розміром 8x8), що представлені(закодовані) у вигляді цілочисельних векторів (1x64), та вектору розмірністю 1x n_samples - вектору класів до якого належить кожен з прикладів. Для вирішення задачі кластеризації вектор класів використовувати не будемо. Графічне представлення - покажемо декілька зображень рукописних цифр(прикладів) які у подальшому будемо кластеризувати. Окрім цього корисно зобразити всі приклади датасету, для візуалізації різниці між прикладами. Оскільки дані багатовимірні, використаємо метод головних компонент(PCA) для зменшення вимірності.

Input:
```
from sklearn import datasets, metrics
import matplotlib.pyplot as plt

digits_X, digits_y = datasets.load_digits(return_X_y=True)

digits_num_clusters = 10

sns.set()
fig = plt.figure()
fig.subplots_adjust()
```

```
for i in range(9):
  ax = fig.add_subplot(3, 3, i + 1, xticks=[], yticks=[])
  ax.imshow(digits_X[i].reshape((8,8)), cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()

pca = PCA(n_components=2)
digits_projection = pca.fit_transform(digits_X)
plt.scatter(digits_projection[:, 0], digits_projection[:, 1])
plt.show()
```

Output:





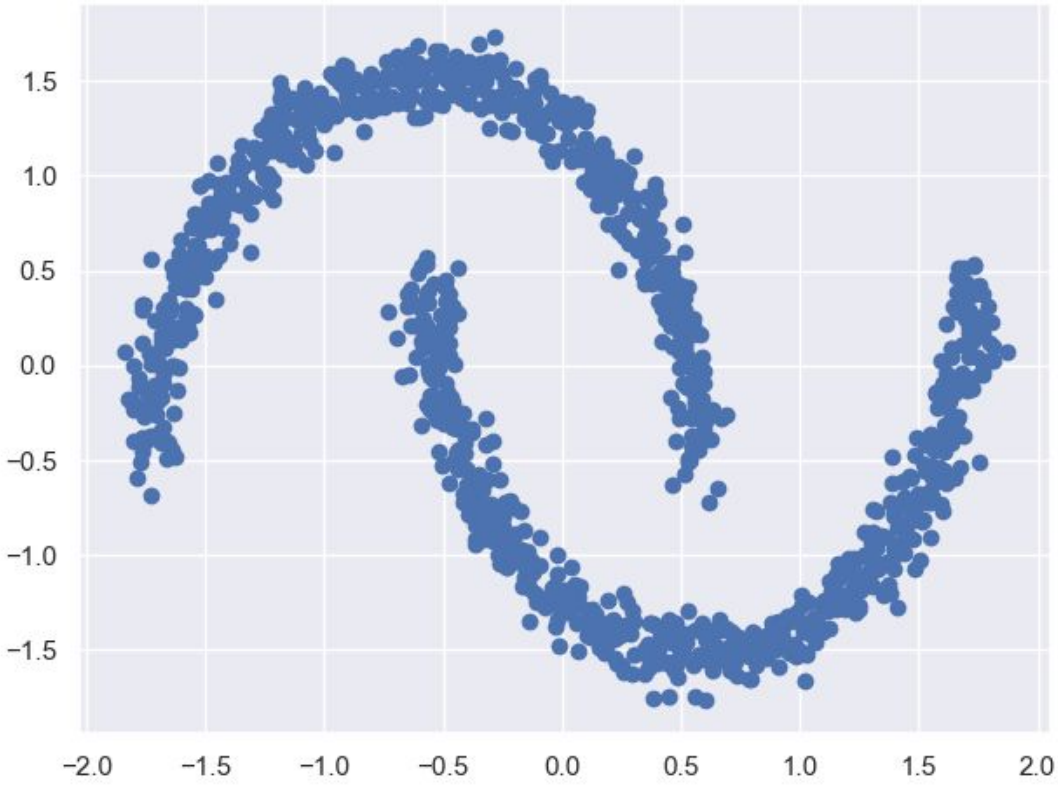
(b)make_moons

Цей датасет складається з набору n_samples двовимірних точок, що утворюють два півкола котрі напів перетинаються, та вектору з 0 та 1 що описує якому півколу(класу) належить точка. Параметр noise описує ступінь розкиданості прикладів в датасеті відносно ліній півкіл. Для задачі кластеризації обираємо невелике занчення noise. Зобразимо точки(приклади) без урахування належності їх конкретному класу.

Input:

```
moons_X, moons_y = datasets.make_moons(n_samples=1200, noise=0.05)
moons_num_clusters = 2

plt.scatter(moons_X[:, 0], moons_X[:, 1])
plt.show()
```

Output:



2. Побудувати модель кластеризації згідно з варіантом.

Будуємо моделі кластеризації за допомогою класу SpectralClustering() модуля sklearn.cluster.

Input:
```
from sklearn.cluster import SpectralClustering
digits_spectral = SpectralClustering(n_clusters=digits_num_clusters,
                                     eigen_solver='arpack',
                                     affinity="nearest_neighbors")
moons_spectral = SpectralClustering(n_clusters=moons_num_clusters,
                                    eigen_solver='arpack',
                                    affinity="nearest_neighbors")
```

3. Виконати кластеризацію даних на основі моделі.

Кластеризацію виконуємо за допомогою методу fit_predict(X) побудованих у попередньому пункті моделей.

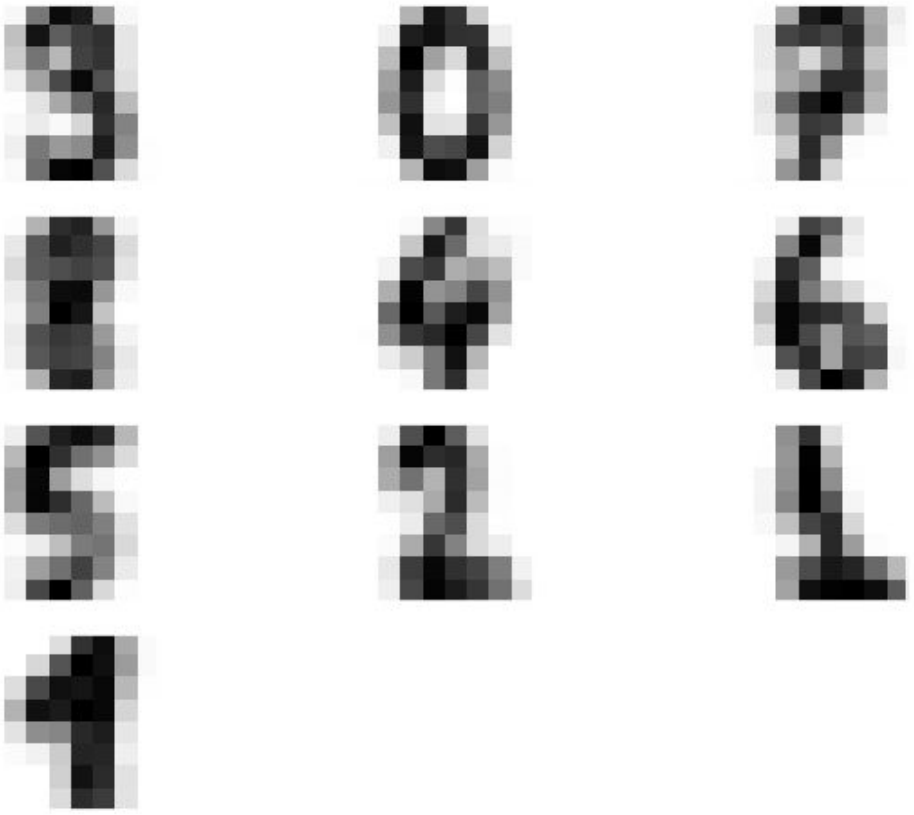
Input:
```
digits_clusters = digits_spectral.fit_predict(digits.data)
moons_clusters = moons_spectral.fit_predict(moons_X)
```

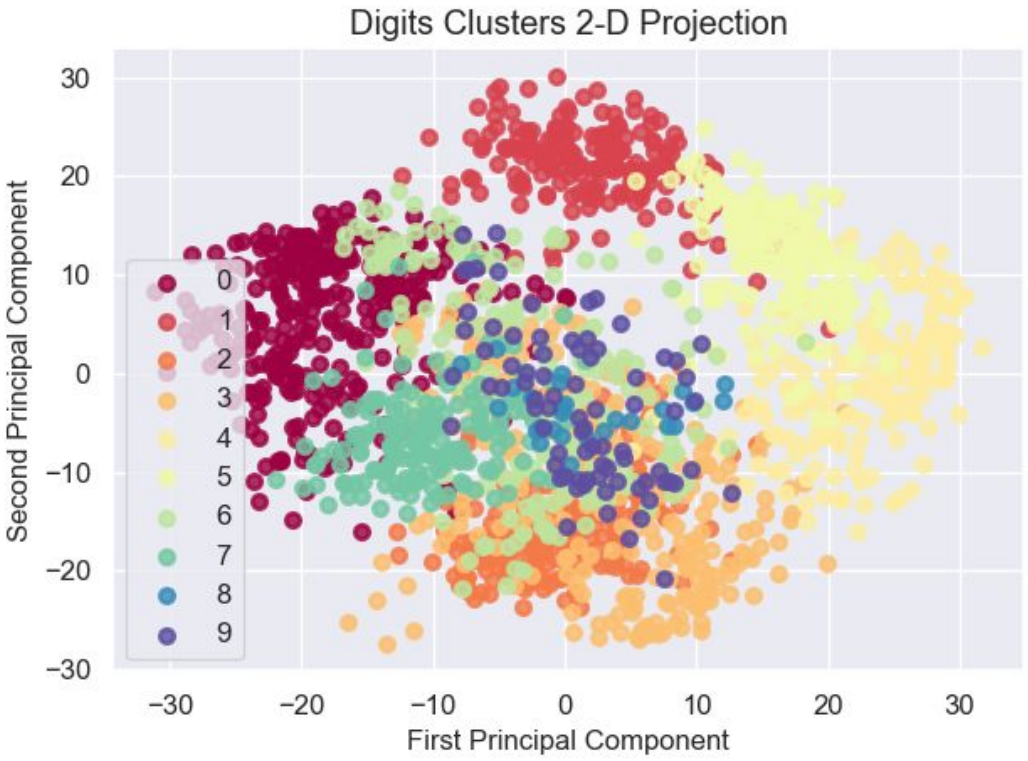
4. Представити розбиття на кластери графічно, наприклад, різними кольо-Рами.

(a)load_digits

В якості графічного представлення розбиття на кластери наведемо зображення центроїда кожного з 10-ти кластерів та двовимірну проекцію усіх прикладів позначених кольором відповідно до кластеру до якого вони належать.

Input:
```
import numpy as np
import scikitplot as skplt

fig = plt.figure()
for c in range(digits_num_clusters):
  ax = fig.add_subplot(4, 3, 1 + c, xticks=[], yticks=[])
  f = digits.data[digits_clusters == c, :]
  cluster_center = np.mean(f, axis=0)
  ax.imshow(cluster_center.reshape((8, 8)), cmap=plt.cm.binary)
plt.show()

skplt.decomposition.plot_pca_2d_projection(pca, digits.data, digits_clusters, title='Digits Clusters 2-D Projection')
plt.show()
```
Output:





5. Розрахувати додаткові результати кластеризації згідно з варіантом.

У варіанті не вказано розраховувати додаткові результати

6. Побудувати декілька альтернативних моделей:

- шляхом зміни значень параметрів основної моделі,
- використати різні функції відстані,

У лінійній алгебрі, **власний розклад** або **спектральний розклад** — це розклад матриці в канонічну форму, таким чином ми представляємо матрицю в термінах її власних значень і власних векторів. Тільки діагональні матриці можна так розкласти. eigen_solver{None, ‘arpack’, ‘lobpcg’, or ‘amg’} - параметр що визначає стратегію спектрального розкладу, AMG використовувати не будемо, оскільки він може призвести до нестабільної роботи, і використання виправдане лише на дуже великих задачах. Для нашої задачі обираємо ‘arpack’.

Матриця спорідненості, яку також називають матрицею подібності, є важливою статистичною технікою, яка використовується для організації взаємної подібності між набором точок даних. Подібність схожа на відстань, однак вона не задовольняє властивостям норми, дві однакові точки будуть мати показник подібності 1, тоді як обчислення норми призведе до нуля.

Привласнюючи числове значення абстрактному поняттю подібності, матриця спорідненості дозволяє програмам машинного навчання імітувати людську логіку, роблячи освічені здогадки про те, як інформація пов’язана та наскільки вона схожа.Що також корисно, матриця подібності дозволяє системам машинного навчання працювати з даними з немаркованих або дещо помилкових наборів даних використовуючи “людський” підхід, що призводить до безлічі практичних застосувань у багатьох сферах. Параметр affinity регулює метод конструювання матриці подібності. На наших даних дослідимо 'rbf', 'nearest_neighbors' 'laplacian', 'cosine' 'linear' ‘poly’

Input:
```
digits_spectrals = []
moons_spectrals = []

print('----------Tunning How to Construct the Affinity Matrix----------')
```

```
for moons_affin, digits_affin in [['rbf', 'nearest_neighbors'],
                                  ['laplacian', 'cosine'],
                                  ['poly', 'poly'],
                                  ['nearest_neighbors', 'linear']]:
  digits_spectrals.append(SpectralClustering(n_clusters=digits_num_clusters,
                                   eigen_solver='arpack',
                                   affinity=digits_affin).fit_predict(digits_X))
  moons_spectrals.append(SpectralClustering(n_clusters=moons_num_clusters,
                                  eigen_solver='arpack',
                                  affinity=moons_affin).fit_predict(moons_X))
```

• задати різні значення кількості кластерів, в алгоритмах де кількість кластерів - параметр.

Input:
```
digits_spectrals.clear()
moons_spectrals.clear()

digits_spectrals = []
moons_spectrals = []
i = 0

print('\n\n----------Tunning Number of Clusters----------\n\n')
for moons_num, digits_num in zip([2, 3, 4, 5, 6], [7, 8, 9, 10, 11]):
  digits_spectrals.append(SpectralClustering(n_clusters=digits_num,
                                   eigen_solver='arpack',
                                   affinity='nearest_neighbors').fit_predict(digits_X))
  moons_spectrals.append(SpectralClustering(n_clusters=moons_num,
                                  eigen_solver='arpack',
                                  affinity='nearest_neighbors').fit_predict(moons_X))
```

7. Для кожної альтернативної моделі розрахувати метрики якості кластеризації, що реалізовані в класі metrics, згідно з варіантом:

• Estimated Number of Clusters.

Для SpectralClustering() кількість кластерів є параметром, а не метрикою якості, тому в цій графі завжди будемо виводити 2 та 10, що є справжньою кількістю класів у датасетах.

• Adjusted Rand Index.

Підраховує міру подібності між еталонною та спрогнозованою кластеризаціями, з модифікацією, що робить значення цього індексу близьким до нуля коли кластери помічені навмання і одиницею коли кластеризаціїї ідентичні. Так як в якості еталонної кластеризації використовуємо справжні класи датасетів, найуспішнішою будемо вважати модель, яка дає значення ближчі до 1.

• Silhouette Coefficient

Середнє значення (b - a) / max(a, b) , де a - середня відстань між прикладами всередині кластеру, b - середня відстань до сусідніх кластерів. a і b підраховуються для кожного прикладу кожного кластеру. Найкраща модель - значення близькі до 1, найгірша -1, значення біля 0 свідчать про те що кластери перетинаються

• Davies-Bouldin index.

Оцінка визначається як середня міра подібності кожного кластера з його найбільш схожим кластером, де схожість - відношення між відстанями всередині кластера та відстанями між кластерами. Таким чином, кластери, які розташовані далі і менш розсіяні, приведуть до кращого(ближчого до 0) балу.

Функція підрахунку метрик:

```
def clustering_metrics(moons_labels, digits_labels):
 data = {'moons': [metrics.adjusted_rand_score(moons_y, moons_labels),
                   metrics.davies_bouldin_score(moons_X, moons_labels),
                   metrics.silhouette_score(moons_X, moons_labels),
                   moons_num_clusters],
         'digits': [metrics.adjusted_rand_score(digits_y, digits_labels),
                    metrics.silhouette_score(digits_X, digits_labels),
                    metrics.davies_bouldin_score(digits_X, digits_labels),
                    digits_num_clusters]}

 spectral_metrics = DataFrame(data=data, index=['Adjusted Rand Index', 'Silhouette coefficient',
                              'Davies-Bouldin index', 'Estimated number of clusters'])
 return spectral_metrics
```

Для того щоб показати метрики для усіх створених моделей, допишемо наступний код всередину циклів, представлених у пункті 6.

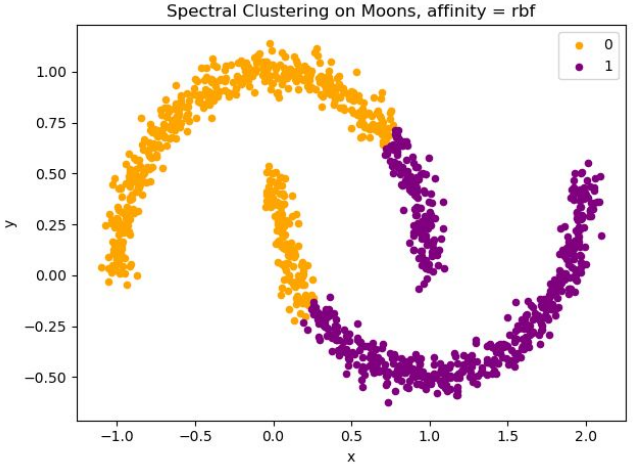
Для першого циклу:

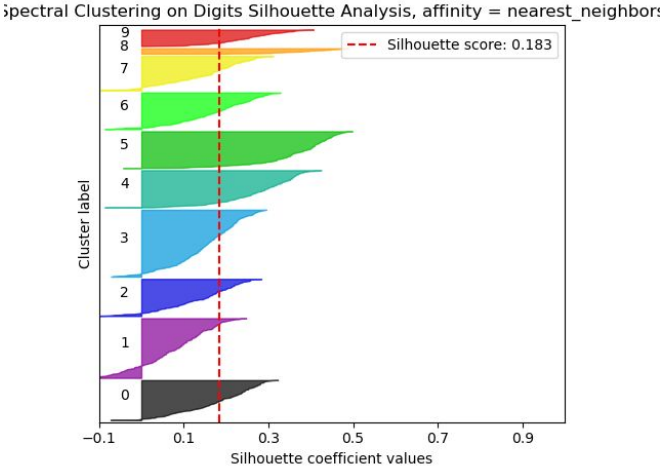
Input:

```
print(f"*****************Metrics of Spectral Clustering**************************\n"
      f"Digits model #{i+1}: \n {digits_num_clusters} clusters, {digits_affin} affinity"
      f"\nMoons model #{i+1}: \n {moons_num_clusters} clusters, {moons_affin} affinity"
      f"\n{clustering_metrics(moons_spectrals[i], digits_spectrals[i])}")
df = DataFrame(dict(x=moons_X[:, 0], y=moons_X[:, 1], label=moons_spectrals[i]))
colors = {0: 'orange', 1: 'purple'}
fig, ax = plt.subplots()
grouped = df.groupby('label')
for key, group in grouped:
  group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key],
          title=f'Spectral Clustering on Moons, affinity = {moons_affin}')
plt.show()
skplt.metrics.plot_silhouette(digits_X, digits_spectrals[i],
                    title=f'Spectral Clustering on Digits Silhouette Analysis, affinity = {digits_affin}')
plt.show()
i += 1
```

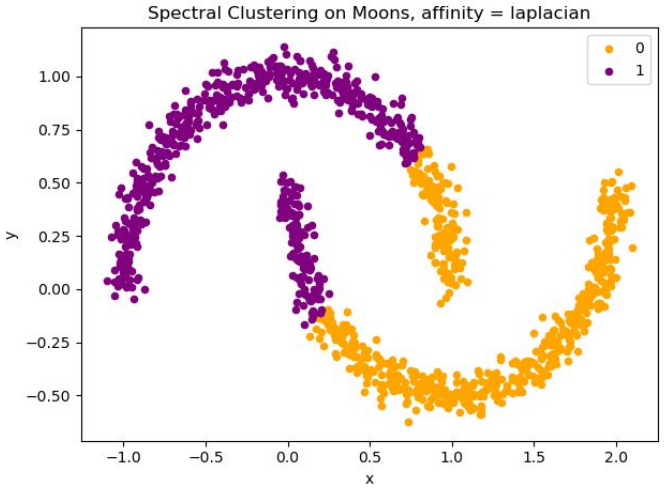
Output:

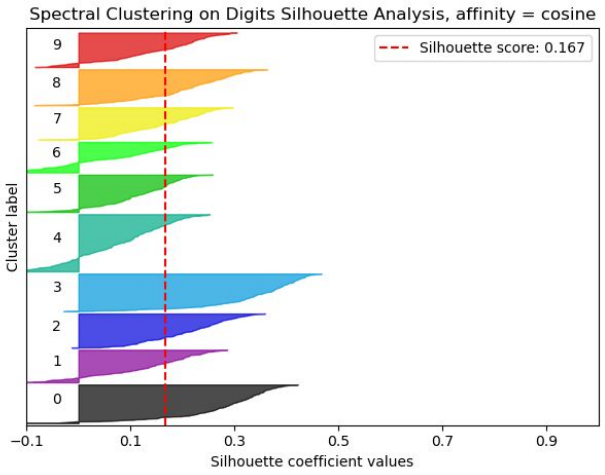
```
--------------Tunning How to Construct the Affinity Matrix-------------------
*****************Metrics of Spectral Clustering*****************************
Digits model #1:
 10 clusters, nearest_neighbors affinity
Moons model #1:
 2 clusters, rbf affinity
                                   moons     digits
Adjusted Rand Index             0.285629   0.756461
Silhouette coefficient          0.780717   0.182729
Davies-Bouldin index            0.491044   1.799007
Estimated number of clusters    2.000000  10.000000
```



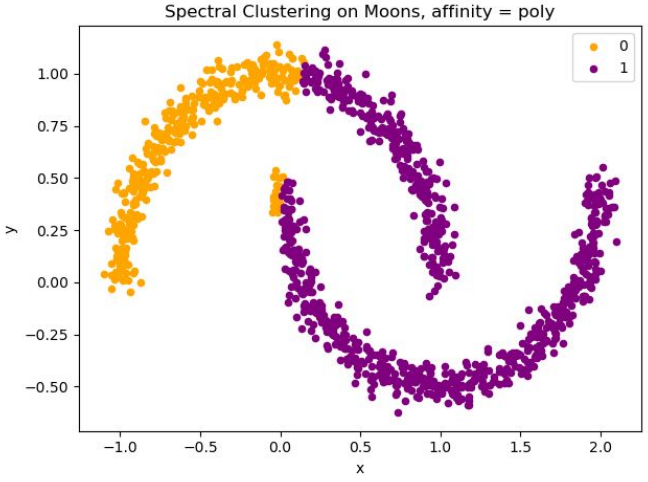


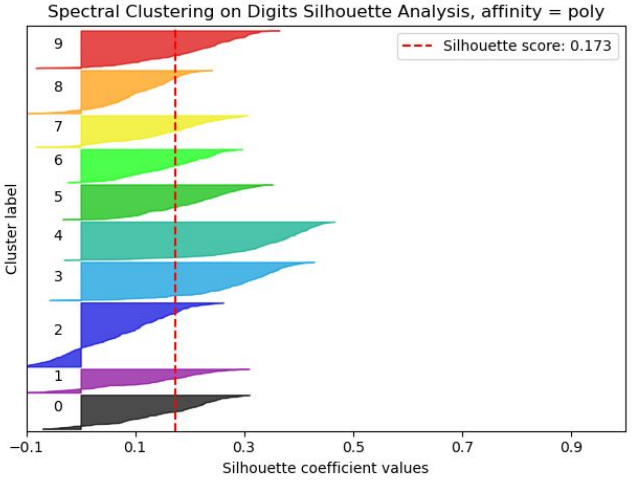
```
*****************Metrics of Spectral Clustering*****************************
Digits model #2:
 10 clusters, cosine affinity
Moons model #2:
 2 clusters, laplacian affinity
                                 moons     digits
Adjusted Rand Index           0.335846   0.630562
Silhouette coefficient        0.785467   0.167205
Davies-Bouldin index          0.489092   1.957825
Estimated number of clusters  2.000000  10.000000
```





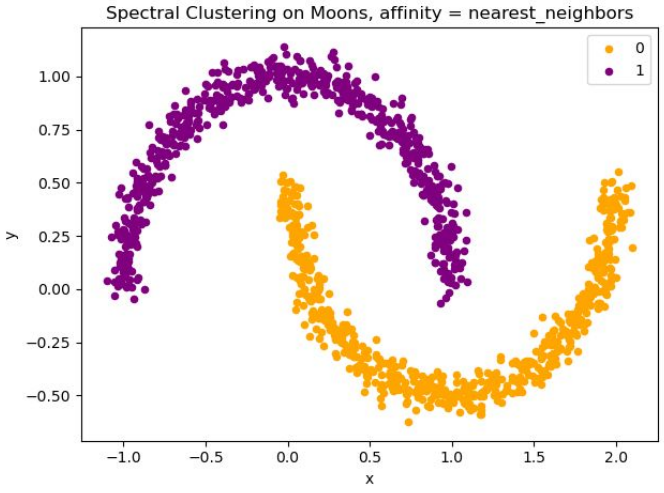

```
*****************Metrics of Spectral Clustering*****************************
Digits model #3:
 10 clusters, poly affinity
Moons model #3:
 2 clusters, poly affinity
                                 moons     digits
Adjusted Rand Index           0.266444   0.639544
Silhouette coefficient        0.731444   0.173495
Davies-Bouldin index          0.444252   1.926702
Estimated number of clusters  2.000000  10.000000
```

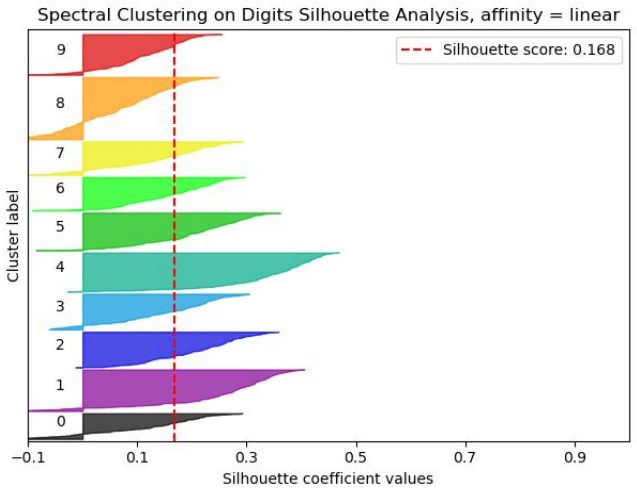





```
  warnings.warn("Graph is not fully connected, spectral embedding"
```

```
**********Metrics of Spectral Clustering**********
Digits model #4:
 10 clusters, linear affinity
Moons model #4:
 2 clusters, nearest_neighbors affinity
                                    moons     digits
Adjusted Rand Index              1.000000   0.629169
Silhouette coefficient           1.146781   0.167646
Davies-Bouldin index             0.336805   1.957517
Estimated number of clusters     2.000000  10.000000
```





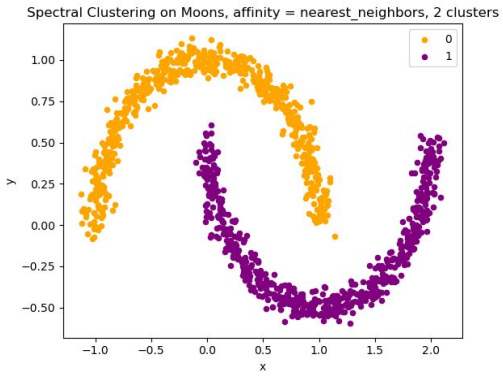
Для другого циклу:

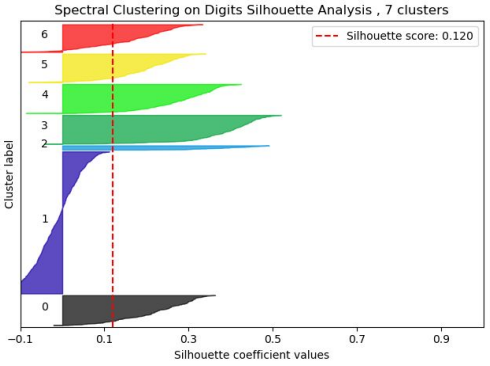
```
Input:  print('\n\n----------Tunning Number of Clusters----------\n\n')
        for moons_num, digits_num in zip([2, 3, 4, 5, 6], [7, 8, 9, 10, 11]):
          digits_spectrals.append(SpectralClustering(n_clusters=digits_num,
                                  eigen_solver='arpack',
                                  affinity='nearest_neighbors').fit_predict(digits_X))
          moons_spectrals.append(SpectralClustering(n_clusters=moons_num,
                                 eigen_solver='arpack',
                                 affinity='nearest_neighbors').fit_predict(moons_X))
          print(f"**********Metrics of Spectral Clustering**********\n"
              f"Digits model #{i+1}: \n {digits_num} clusters, nearest_neighbors affinity"
              f"\nMoons model #{i+1}: \n {moons_num} clusters, nearest_neighbors affinity"
              f"\n{clustering_metrics(moons_spectrals[i], digits_spectrals[i])}")
          df = DataFrame(dict(x=moons_X[:, 0], y=moons_X[:, 1], label=moons_spectrals[i]))
          colors = {0: 'orange', 1: 'purple', 2: 'cyan', 3: 'yellow', 4: 'brown', 5: 'blue'}
          fig, ax = plt.subplots()
          grouped = df.groupby('label')
          for key, group in grouped:
             group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key],
                    title=f'Spectral Clustering on Moons, affinity = nearest_neighbors, {moons_num} clusters')
          plt.show()
          skplt.metrics.plot_silhouette(digits_X, digits_spectrals[i],
                                title=f'Spectral Clustering on Digits Silhouette Analysis , '
                                    f'{digits_num} clusters')
          skplt.metrics.plot_silhouette(moons_X, moons_spectrals[i],
                                title=f'Spectral Clustering on Moons Silhouette Analysis , '
```

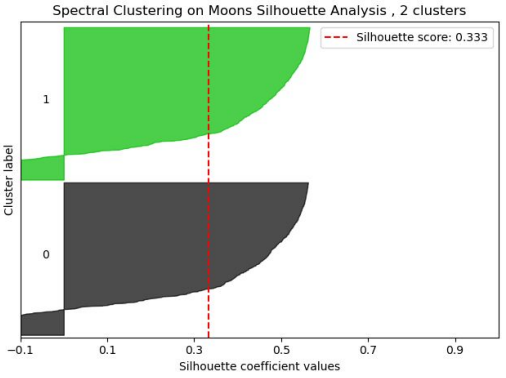
```
                        f'{moons_num} clusters')
    i += 1
plt.show()
digits_spectrals.clear()
moons_spectrals.clear()
```

Output:

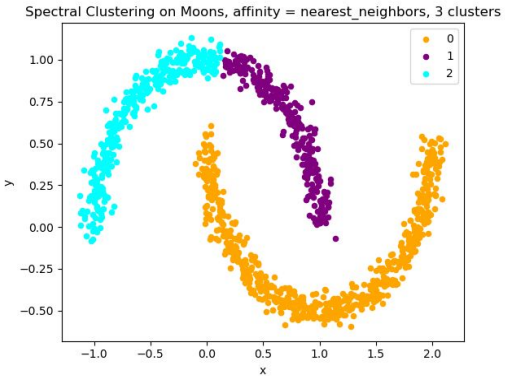
```
--------------Tunning Number of Clusters-------------------


*****************Metrics of Spectral Clustering*****************************
Digits model #1:
 7 clusters, nearest_neighbors affinity
Moons model #1:
 2 clusters, nearest_neighbors affinity
                                  moons     digits
Adjusted Rand Index           1.000000   0.403180
Silhouette coefficient        1.159658   0.120050
Davies-Bouldin index          0.332860   1.948665
Estimated number of clusters  2.000000  10.000000
```

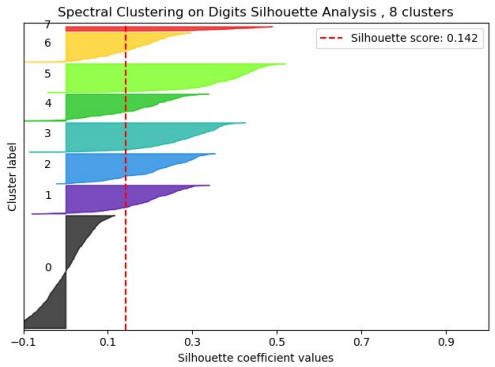


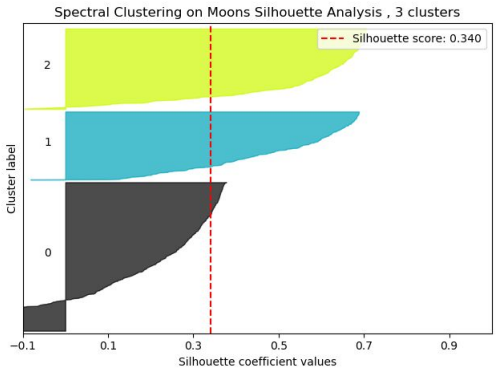





```
*****************Metrics of Spectral Clustering*****************************
Digits model #2:
 8 clusters, nearest_neighbors affinity
Moons model #2:
 3 clusters, nearest_neighbors affinity
                                  moons     digits
Adjusted Rand Index           0.751477   0.547733
Silhouette coefficient        1.131833   0.141500
Davies-Bouldin index          0.340197   1.947305
Estimated number of clusters  2.000000  10.000000
```



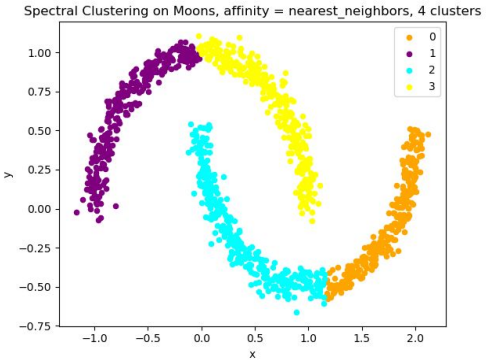


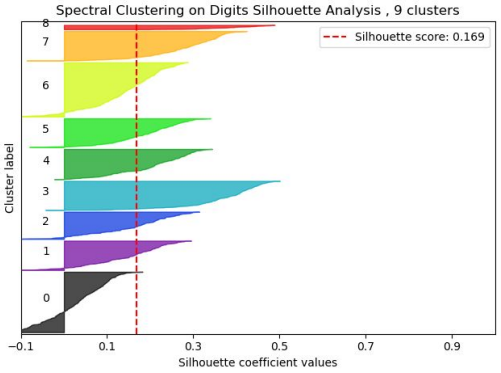


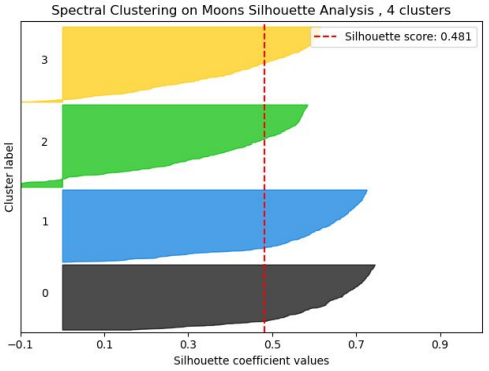
**********Metrics of Spectral Clustering**********

Digits model #3:
 9 clusters, nearest_neighbors affinity
Moons model #3:
 4 clusters, nearest_neighbors affinity

	moons	digits
Adjusted Rand Index	0.501113	0.743701
Silhouette coefficient	0.794483	0.168937
Davies-Bouldin index	0.486871	1.910498
Estimated number of clusters	2.000000	10.000000



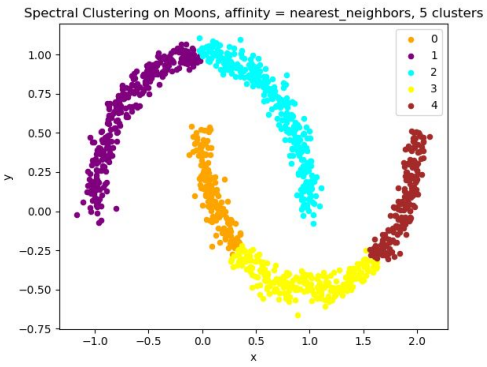


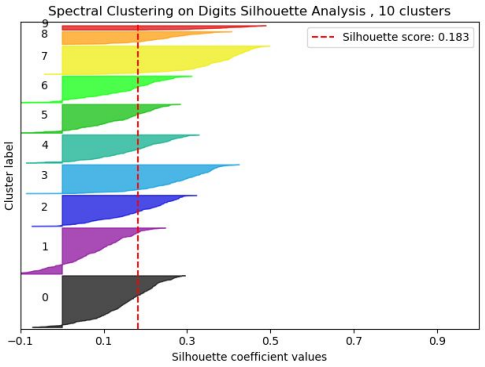


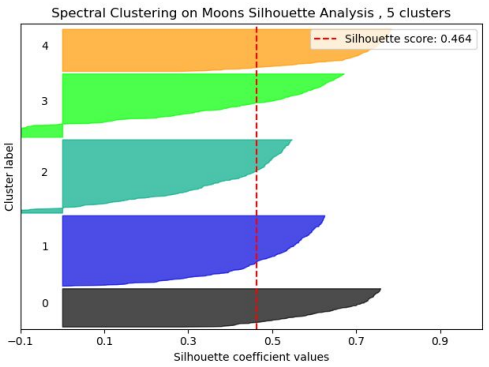
**********Metrics of Spectral Clustering**********

Digits model #4:
 10 clusters, nearest_neighbors affinity
Moons model #4:
 5 clusters, nearest_neighbors affinity

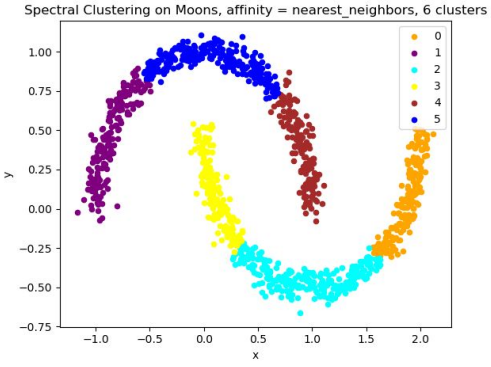
	moons	digits
Adjusted Rand Index	0.422983	0.756461
Silhouette coefficient	0.714673	0.182729
Davies-Bouldin index	0.470670	1.799007
Estimated number of clusters	2.000000	10.000000

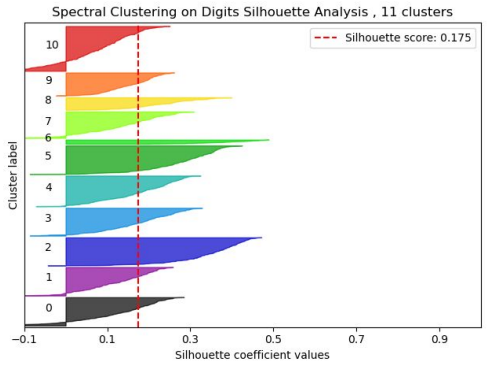


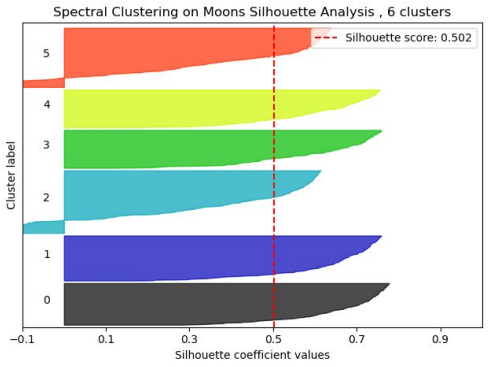





```
**********Metrics of Spectral Clustering**********
Digits model #5:
 11 clusters, nearest_neighbors affinity
Moons model #5:
 6 clusters, nearest_neighbors affinity
                                 moons     digits
Adjusted Rand Index           0.353644   0.836204
Silhouette coefficient        0.653393   0.174738
Davies-Bouldin index          0.490727   1.888674
Estimated number of clusters  2.000000  10.000000
```







8. Виконати аналіз результатів кластеризації одним з неформальних методів згідно з варіантом:

Чи є розбиття стабільним після видалення окремих об'єктів?

Спочатку наведемо метрики, розмірність даних та візуалізацію кластерів до видалення окремих елементів:

Input:
```
print(f'Shapes before deletion of elements:\n'
      f'  moons_X: {moons_X.shape}\n'
      f'  moons_y: {moons_y.shape}\n'
      f'  digits_X: {digits_X.shape}\n'
      f'  digits_y: {digits_y.shape}\n')
print(f'Metrics before deletion:\n{clustering_metrics(moons_clusters, digits_clusters)}')
plt.scatter(moons_X[:, 0], moons_X[:, 1], c=moons_clusters, cmap='PuOr')
plt.show()
```

```
skplt.decomposition.plot_pca_2d_projection(pca, digits_X, digits_clusters,
                                           title='Digits Clusters 2-D Projection Before Deletion')
plt.show()

fig = plt.figure()
fig.suptitle('Cluster Centres Before Deletion')
for c in range(digits_num_clusters):
  ax = fig.add_subplot(4, 3, 1 + c, xticks=[], yticks=[])
  f = digits_X[digits_clusters == c, :]
  cluster_center = np.mean(f, axis=0)
  ax.imshow(cluster_center.reshape((8, 8)), cmap=plt.cm.binary)
plt.show()
```

Output:

```
Shapes before deletion of elements:
   moons_X: (1200, 2)
   moons_y: (1200,)
   digits_X: (1797, 64)
   digits_y: (1797,)

Metrics before deletion:
                                  moons     digits
Adjusted Rand Index            1.000000   0.756461
Silhouette coefficient         1.152948   0.182729
Davies-Bouldin index           0.335111   1.799007
Estimated number of clusters   2.000000  10.000000
```

Тепер видалимо по 500 випадкових об’єктів з кожного датасету і порівняємо результати графіки перед і після видалення покажемо разом для наочності:

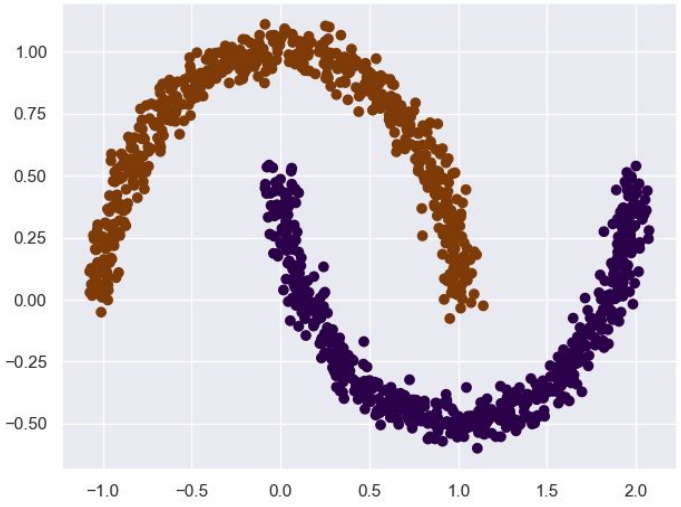
Input:

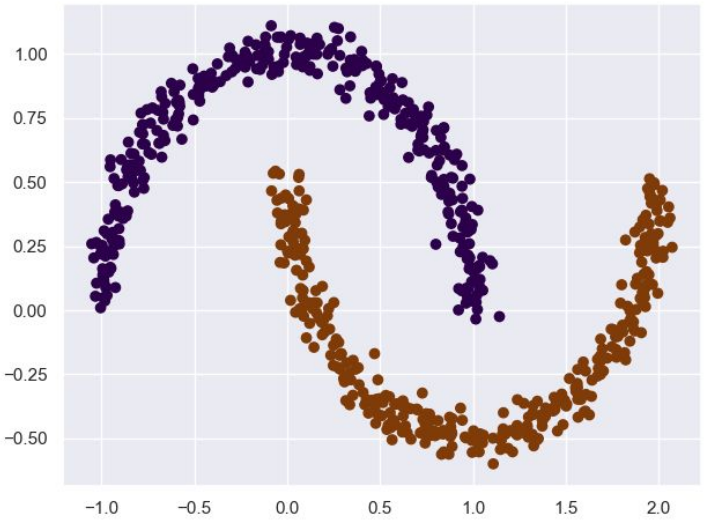
```
count = 0
for count in range(500):
  deletion_position_moons = np.random.randint(0, len(moons_X))
  deletion_position_digits = np.random.randint(0, len(digits_X))
  digits_X = np.delete(digits_X, deletion_position_digits, axis=0)
  moons_X = np.delete(moons_X, deletion_position_moons, axis=0)
  digits_y = np.delete(digits_y, deletion_position_digits, axis=0)
  moons_y = np.delete(moons_y, deletion_position_moons, axis=0)

print(f'Shapes after deletion of elements:\n'
   f'  moons_X: {moons_X.shape}\n'
   f'  moons_y: {moons_y.shape}\n'
   f'  digits_X: {digits_X.shape}\n'
   f'  digits_y: {digits_y.shape}\n')

digits_clusters = digits_spectral.fit_predict(digits_X)
```

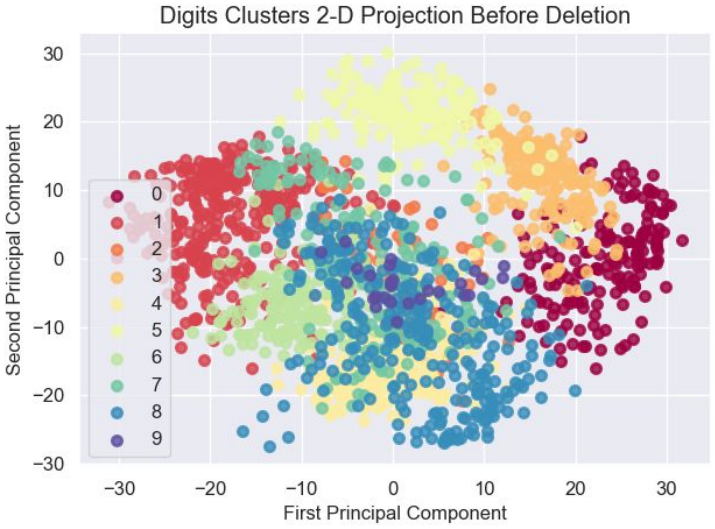
```
moons_clusters = moons_spectral.fit_predict(moons_X)

plt.scatter(moons_X[:, 0], moons_X[:, 1], c=moons_clusters, cmap='PuOr')
plt.show()
skplt.decomposition.plot_pca_2d_projection(pca, digits_X, digits_clusters, title='Digits Clusters 2-D Projection After Deletion')
plt.show()
print(f'Metrics after deletion:\n{clustering_metrics(moons_clusters, digits_clusters)}')

fig = plt.figure()
fig.suptitle('Cluster Centres After Deletion')
for c in range(digits_num_clusters):
  ax = fig.add_subplot(4, 3, 1 + c, xticks=[], yticks=[])
  f = digits_X[digits_clusters == c, :]
  cluster_center = np.mean(f, axis=0)
  ax.imshow(cluster_center.reshape((8, 8)), cmap=plt.cm.binary)
plt.show()
```

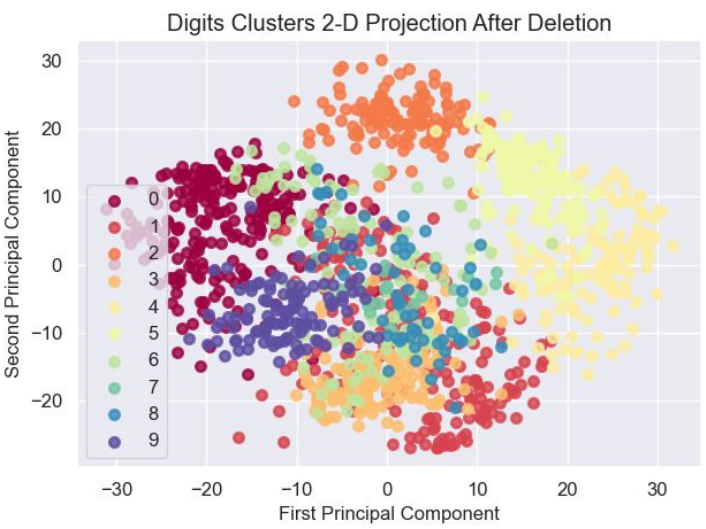
Output:

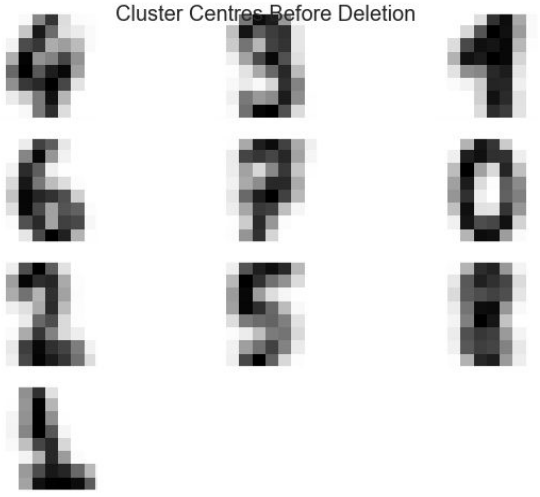
```
Shapes after deletion of elements:
   moons_X: (700, 2)
   moons_y: (700,)
   digits_X: (1297, 64)
   digits_y: (1297,)

Metrics after deletion:
                                  moons     digits
Adjusted Rand Index            1.000000   0.754105
Silhouette coefficient         1.208701   0.188306
Davies-Bouldin index           0.317763   1.764396
Estimated number of clusters   2.000000  10.000000
```

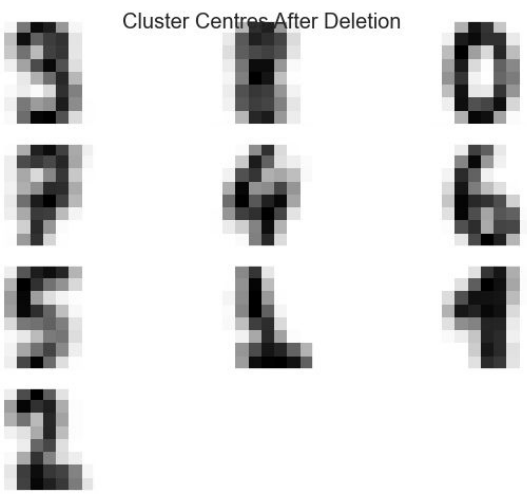












9. Зробити висновки про якість роботи моделей на досліджених даних. Дослідити різні значення параметрів основної моделі, різні функції відстані та різну кількість кластерів в алгоритмах, де кількість кластерів слугує параметром.

На make_moons SpectralClustering() працює відмінно з правильно підібраними параметрами. На load_digits гірше, це пояснюється близькістю та інодф навіть перетином істиних кластерів у цьому наборі даних. Функції відстані не використовуються, використовуються так звані функції подібності, різниця була пояснена у п.6. Найгірше на make_moons працює poly affinity, на load_digits - linear affinity, найкраще на обох моделях працює nearest_neighbors affinity. Щодо кількості кластерів - для

нашої задачі слугує параметром, і здається інтуїтивно зрозумілим,що найкраща кількість кластерів - та що справді є у датасетів. Експериментально підтвердили це.

10. Оцінити результати кластеризації на основі метрик якості та на основі неформальних методів. Для кожного набору даних вибрати найкращу модель.

Найкращі моделі - потачкові

```
digits_spectral = SpectralClustering(n_clusters=digits_num_clusters,
                                     eigen_solver='arpack',
                                     affinity="nearest_neighbors")
moons_spectral = SpectralClustering(n_clusters=moons_num_clusters,
                                    eigen_solver='arpack',
                                affinity="nearest_neighbors")
```

їх метрики:

	moons	digits
Adjusted Rand Index	1.000000	0.756461
Silhouette coefficient	1.152948	0.182729
Davies-Bouldin index	0.335111	1.799007
Estimated number of clusters	2.000000	10.000000

Як бачимо на Make_moons спектральна кластеризація працює майже ідеально. `Adjusted Rand Index` приймає найкраще з можливих значенння модель безпомилково розподілила дані по кластерам, `Silhouette coefficient` на 0.15 перевищує оптимальне значення, що свідчить про те що кластери навіть більш відділені ніж необхідно. індекс Девіса-Болдіна на 0.33 більше за 0 - оптимальне значення, що означає що кластери не є подібними . З load_digits - гірше, оскільки самі вхідні дані не так добре сегментовані, наприклад 1 деякі люди пишуть дуже схожою на 2, для алгоритму кластеризації це перепона. Тому помилки при присвоєнні маркування тут більш суттєві, `Adjusted Rand Index` приймає значення 0.75 що доволі непогано враховуючи те що ми кластеризуємо рукописні цифри. `Silhouette coefficient` близький до нуля, що означає що дані з різних кластерів знаходяться дуже близько один до одного(кластери перетинаються). Індекс Девіса-Болдіна також дуже високий, що знову ж таки свідчить про “погану” з погляду сегментованості структуру вхідних даних.

Після видалення 500 елементів з кожного датасету метрики не зазнали особливих змін, як і структура класів, що свідчить про стабільність моделі після видалення окремих елементів.