МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
Національний технічний університет   
«Харківський політехнічний інститут»  
Кафедра ГМКГ

**Лабораторна работа №4**

З дисципліни «Інтелектуальний аналіз даних»

Виконав:

Студент групи ІКМ-220 г.

Ульянов Кирило Юрійович

Перевірив:

Доц. Дашкевич А.О.

Харків 2023

**Мета роботи:** : вивчення базових алгоритмiв кластеризацiї щiльнiсного та графового типу.

**Завдання на роботу:** завантаження набору даних, формування вхiдної вибiрки даних, кластеризацiя iз застосуванням алгоритмiв DBSCAN та Affinity Propagation, порiвняльний аналiз алгоритмiв кластеризацiї.

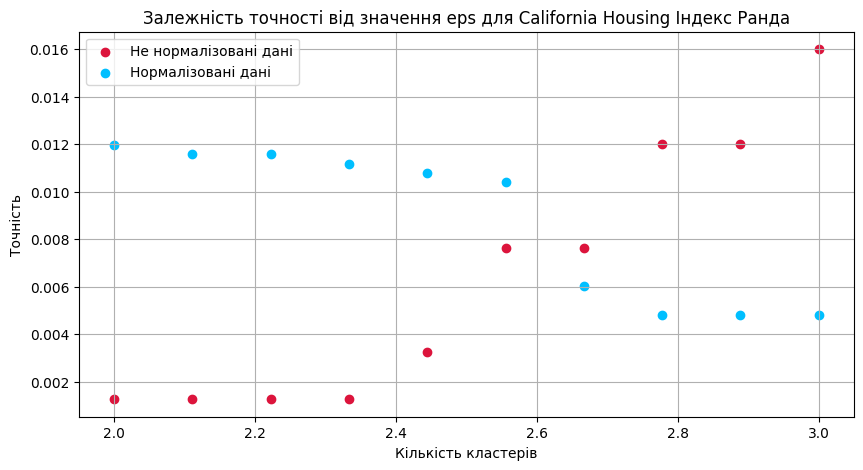
Завантаживши набір даних "California Housing" без класових міток, я взяв 5000 об'єктів для навчання. Також я стандартизував вхідні ознаки.



**Кластеризація методом DBSCAN в залежності від параметру eps**

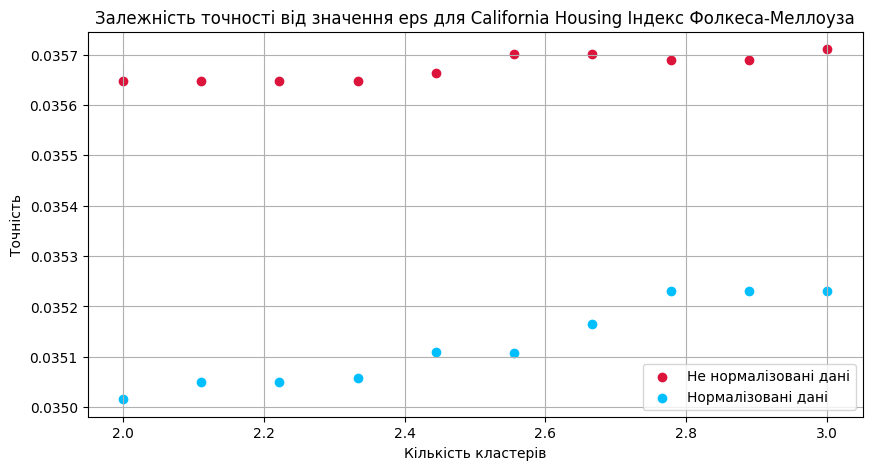
*Залежність точності від значення параметру eps в алгоритмі DBSCAN. Метрика – Індекс Ранда.*

|  |  |  |
| --- | --- | --- |
| Значення eps | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 2.00000 | 0.00127 | 0.01197 |
| 2.11111 | 0.00127 | 0.01157 |
| 2.22222 | 0.00127 | 0.01158 |
| 2.33333 | 0.00127 | 0.01118 |
| 2.44444 | 0.00327 | 0.01079 |
| 2.55556 | 0.00765 | 0.01039 |
| 2.66667 | 0.00765 | 0.00602 |
| 2.77778 | 0.01203 | 0.00483 |
| 2.88889 | 0.01203 | 0.00483 |
| 3.00000 | 0.01600 | 0.00483 |



*Залежність точності від значення параметру eps в алгоритмі DBSCAN. Метрика – Індекс Фолкеса-Меллоуза.*

|  |  |  |
| --- | --- | --- |
| Значення eps | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 2.00000 | 0.03565 | 0.03502 |
| 2.11111 | 0.03565 | 0.03505 |
| 2.22222 | 0.03565 | 0.03505 |
| 2.33333 | 0.03565 | 0.03506 |
| 2.44444 | 0.03566 | 0.03511 |
| 2.55556 | 0.03570 | 0.03511 |
| 2.66667 | 0.03570 | 0.03516 |
| 2.77778 | 0.03569 | 0.03523 |
| 2.88889 | 0.03569 | 0.03523 |
| 3.00000 | 0.03571 | 0.03523 |

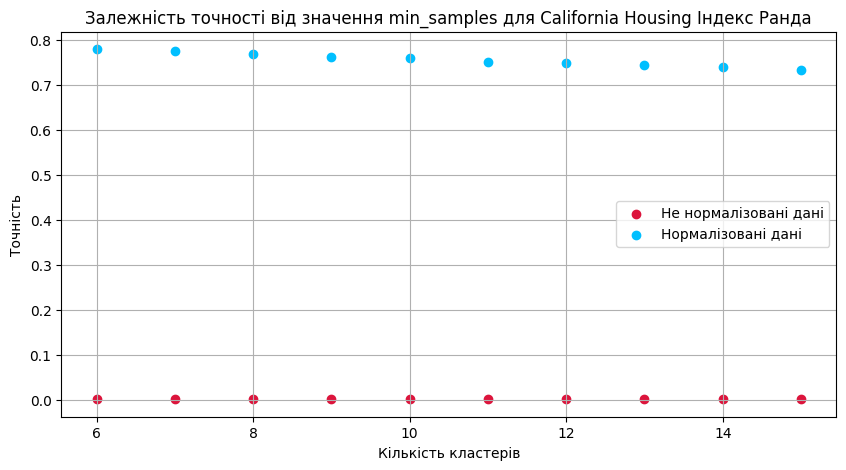


**Кластеризація методом DBSCAN в залежності від параметру *min\_samples***

*Залежність точності від значення параметру min\_samples в алгоритм DBSCAN.*

*Метрика – Індекс Ранда.*

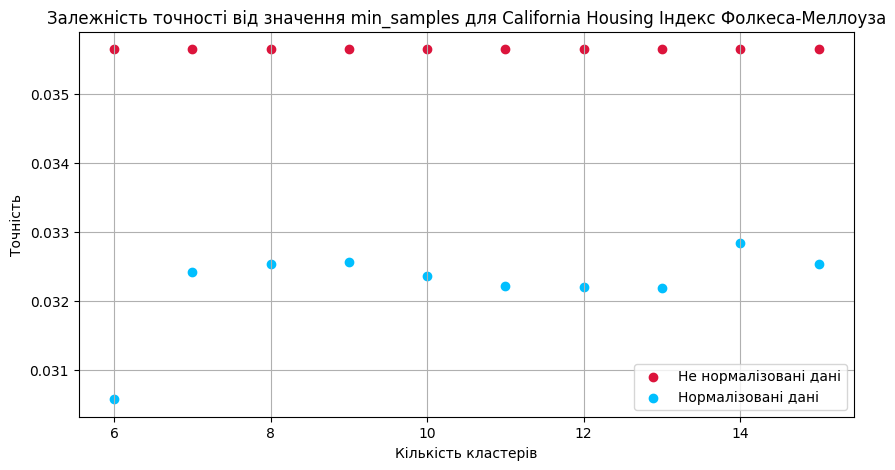
|  |  |  |
| --- | --- | --- |
| Значення min\_samples | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 6 | 0.00127 | 0.77970 |
| 7 | 0.00127 | 0.77475 |
| 8 | 0.00127 | 0.76906 |
| 9 | 0.00127 | 0.76230 |
| 10 | 0.00127 | 0.75976 |
| 11 | 0.00127 | 0.75198 |
| 12 | 0.00127 | 0.74896 |
| 13 | 0.00127 | 0.74376 |
| 14 | 0.00127 | 0.74032 |
| 15 | 0.00127 | 0.73355 |



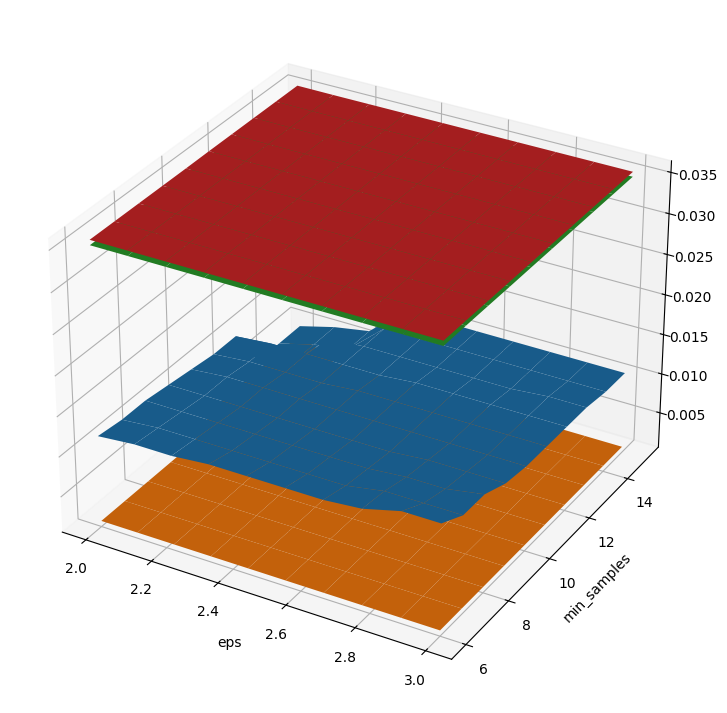
*Залежність точності від значення параметру min\_samples в алгоритмі DBSCAN.*

*Метрика – Індекс Фолкеса-Меллоуза.*

|  |  |  |
| --- | --- | --- |
| Значення min\_samples | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 6 | 0.03565 | 0.03057 |
| 7 | 0.03565 | 0.03241 |
| 8 | 0.03565 | 0.03253 |
| 9 | 0.03565 | 0.03256 |
| 10 | 0.03565 | 0.03236 |
| 11 | 0.03565 | 0.03221 |
| 12 | 0.03565 | 0.03220 |
| 13 | 0.03565 | 0.03219 |
| 14 | 0.03565 | 0.03284 |
| 15 | 0.03565 | 0.03253 |



*Загальний 3д графік для DBSCAN в залежності від min\_samples та eps*

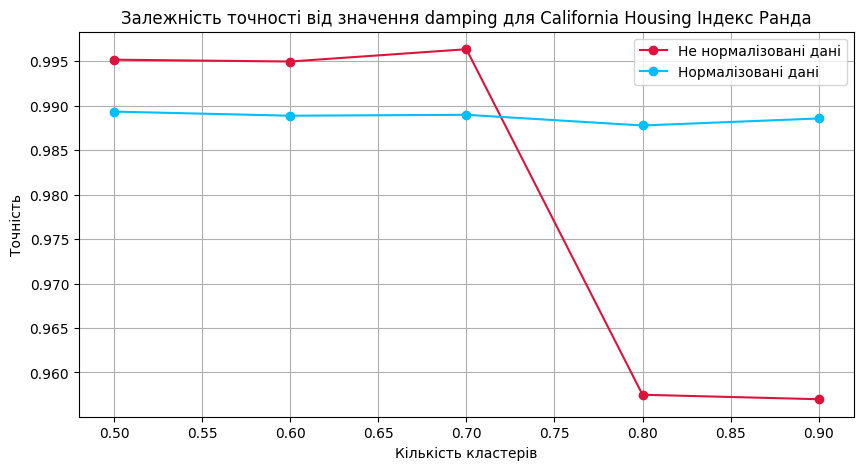


**Кластеризація методом Affinity Propagation в залежності від параметру damping\_range**

*Залежність точності від значення параметру damping в алгоритмі Affinity Propagation.*

*Метрика – Індекс Ранда.*

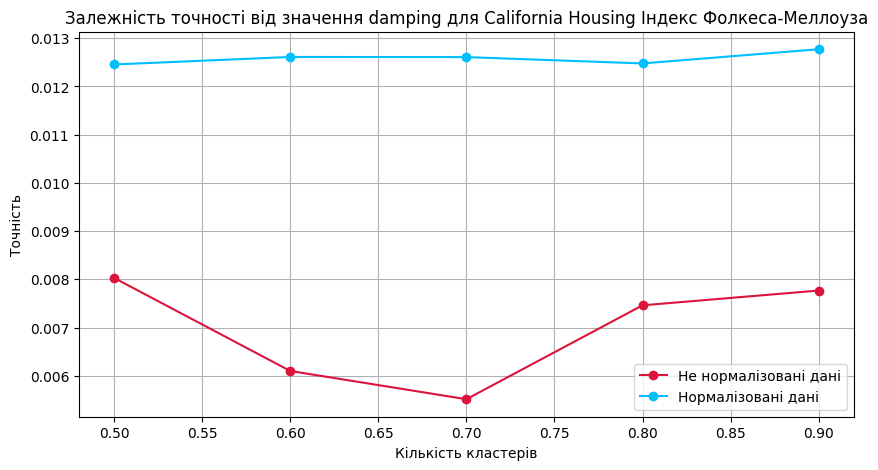
|  |  |  |
| --- | --- | --- |
| Значення damping | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 0.5 | 0.99518 | 0.98935 |
| 0.6 | 0.99498 | 0.98888 |
| 0.7 | 0.99636 | 0.98899 |
| 0.8 | 0.95749 | 0.98778 |
| 0.9 | 0.95699 | 0.98857 |



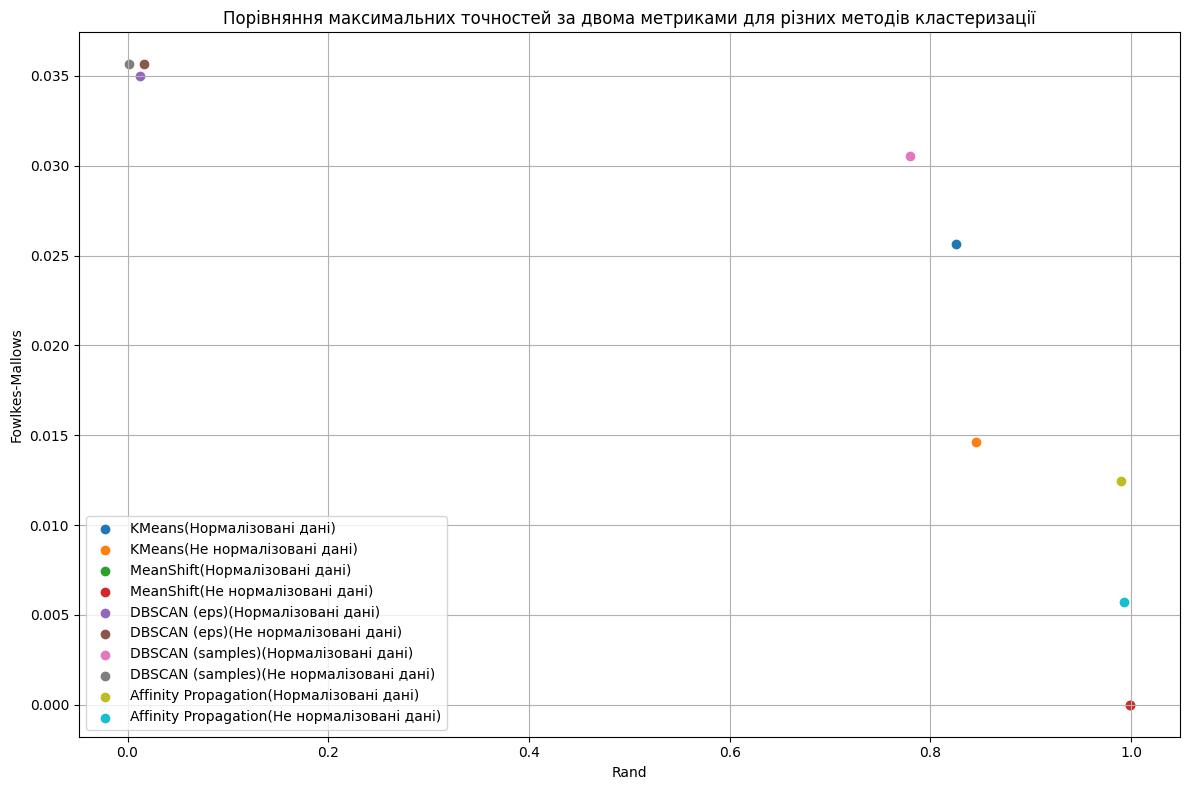
*Залежність точності від значення параметру damping в алгоритмі Affinity Propagation.*

*Метрика – Індекс Фолкеса-Меллоуза.*

|  |  |  |
| --- | --- | --- |
| Значення damping | Точність на ненормалізованих даних | Точність на нормалізованих даних |
| 0.5 | 0.00803 | 0.01246 |
| 0.6 | 0.00610 | 0.01261 |
| 0.7 | 0.00551 | 0.01261 |
| 0.8 | 0.00746 | 0.01248 |
| 0.9 | 0.00777 | 0.01277 |



**Порiвняльна дiаграма алгоритмiв кластеризацiї**



**Код програми:**

**from sklearn.cluster import DBSCAN, AffinityPropagation, KMeans, MeanShift**

**import numpy as np**

**import matplotlib.pyplot as plt**

**from sklearn.preprocessing import StandardScaler, Normalizer**

**from sklearn.datasets import fetch\_california\_housing**

**from sklearn.metrics import rand\_score, fowlkes\_mallows\_score**

**def plot\_metrics(cluster\_range, metric\_list\_unnormalized, metric\_list\_normalized, title=None, param\_name=None, scatter=False):**

**"""**

**Визуализирует зависимость метрик от числа кластеров для необработанных и нормализованных данных.**

**Параметры:**

**- `cluster\_range (range)`: Диапазон числа кластеров.**

**- `metric\_list\_unnormalized (list)`: Список метрик для необработанных данных.**

**- `metric\_list\_normalized (list)`: Список метрик для нормализованных данных.**

**- `title (str)`: Заголовок графика.**

**- `scatter (boolean)`: Вивести точковий графік.**

**Возвращает:**

**- `None`**

**"""**

**if scatter:**

**plt.figure(figsize=(10, 5))**

**plt.scatter(cluster\_range, metric\_list\_unnormalized, marker='o', label='Не нормалізовані дані', color='crimson')**

**plt.scatter(cluster\_range, metric\_list\_normalized, marker='o', label='Нормалізовані дані', color='deepskyblue')**

**plt.xlabel(param\_name)**

**plt.ylabel('Точність')**

**plt.title(title)**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**else:**

**plt.figure(figsize=(10, 5))**

**plt.plot(cluster\_range, metric\_list\_unnormalized, marker='o', label='Не нормалізовані дані', color='crimson')**

**plt.plot(cluster\_range, metric\_list\_normalized, marker='o', label='Нормалізовані дані', color='deepskyblue')**

**plt.xlabel(param\_name)**

**plt.ylabel('Точність')**

**plt.title(title)**

**plt.legend()**

**plt.grid(True)**

**plt.show()**

**california = fetch\_california\_housing()**

**X = california.data[:5000]**

**X\_scaled = StandardScaler().fit\_transform(X)**

***# параметри для кластеризации***

**cluster\_range = range(2, 11)**

***# массивы для записи точностей***

**kmeans\_accuracy\_list\_unnormalized\_rand = []**

**kmeans\_accuracy\_list\_normalized\_rand = []**

**kmeans\_accuracy\_list\_unnormalized\_fowlkes = []**

**kmeans\_accuracy\_list\_normalized\_fowlkes = []**

**for cluster in cluster\_range:**

**kmeans = KMeans(n\_clusters=cluster)**

**predictions = kmeans.fit\_predict(X)**

**accuracy\_rand = rand\_score(california.target[:5000], predictions)**

**kmeans\_accuracy\_list\_unnormalized\_rand.append(accuracy\_rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**kmeans\_accuracy\_list\_unnormalized\_fowlkes.append(accuracy\_fowlkes)**

**kmeans = KMeans(n\_clusters=cluster)**

**predictions = kmeans.fit\_predict(X\_scaled)**

**accuracy\_rand = rand\_score(california.target[:5000], predictions)**

**kmeans\_accuracy\_list\_normalized\_rand.append(accuracy\_rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**kmeans\_accuracy\_list\_normalized\_fowlkes.append(accuracy\_fowlkes)**

***# параметри для кластеризации***

**bandwidth\_range = np.linspace(0.1, 0.5, num=10)**

***# массивы для записи точностей***

**meanshift\_accuracy\_unnormalized\_rand = []**

**meanshift\_accuracy\_normalized\_rand = []**

**meanshift\_accuracy\_unnormalized\_fowlkes = []**

**meanshift\_accuracy\_normalized\_fowlkes = []**

**for bandwidth in bandwidth\_range:**

**mean\_shift = MeanShift(bandwidth=bandwidth)**

**predictions = mean\_shift.fit\_predict(X)**

**accuracy = rand\_score(california.target[:5000], predictions)**

**meanshift\_accuracy\_unnormalized\_rand.append(accuracy)**

**meanshift\_accuracy\_unnormalized\_fowlkes.append(fowlkes\_mallows\_score(california.target[:5000], predictions))**

**mean\_shift = MeanShift(bandwidth=bandwidth)**

**predictions = mean\_shift.fit\_predict(X\_scaled)**

**accuracy = rand\_score(california.target[:5000], predictions)**

**meanshift\_accuracy\_normalized\_rand.append(accuracy)**

**meanshift\_accuracy\_normalized\_fowlkes.append(fowlkes\_mallows\_score(california.target[:5000], predictions))**

***# разные параметры окресности eps\_list***

**eps\_list = np.linspace(2, 3, 10)**

***# минимальное количество точек, необходимых для формирования плотного кластера***

**min\_samples\_list = np.arange(6, 16)**

**dbscan\_unnormalized\_rand = np.empty((eps\_list.shape[0], min\_samples\_list.shape[0]))**

**dbscan\_normalized\_rand = np.empty((eps\_list.shape[0], min\_samples\_list.shape[0]))**

**dbscan\_unnormalized\_fowlkes = np.empty((eps\_list.shape[0], min\_samples\_list.shape[0]))**

**dbscan\_normalized\_fowlkes = np.empty((eps\_list.shape[0], min\_samples\_list.shape[0]))**

**for i, eps in enumerate(eps\_list):**

**for j, sample in enumerate(min\_samples\_list):**

**dbscan = DBSCAN(eps=eps, min\_samples=sample)**

**predictions = dbscan.fit\_predict(X)**

**rand = rand\_score(california.target[:5000], predictions)**

**fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_unnormalized\_rand[i][j] = rand**

**dbscan\_unnormalized\_fowlkes[i][j] = fowlkes**

**dbscan = DBSCAN(eps=eps, min\_samples=sample)**

**predictions = dbscan.fit\_predict(X\_scaled)**

**rand = rand\_score(california.target[:5000], predictions)**

**fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_normalized\_rand[i][j] = rand**

**dbscan\_normalized\_fowlkes[i][j] = fowlkes**

**x,y = np.meshgrid(eps\_list, min\_samples\_list)**

***# Creating figure***

**fig = plt.figure(figsize =(14, 9))**

**ax = plt.axes(projection ='3d')**

***# Creating plot***

**ax.plot\_surface(x, y, dbscan\_normalized\_rand, label = "DBSCAN Normilized Rand")**

**ax.plot\_surface(x, y, dbscan\_unnormalized\_rand)**

**ax.plot\_surface(x, y, dbscan\_normalized\_fowlkes)**

**ax.plot\_surface(x, y, dbscan\_unnormalized\_fowlkes)**

**ax.set\_xlabel('eps')**

**ax.set\_ylabel('min\_samples')**

***# show plot***

**plt.show()**

**pyplot = lambda x, y, z, color: plt.scatter(x,y, s=(z/np.linalg.norm(z))\*300, alpha=0.7, c = color)**

**pyplot(x, y, dbscan\_normalized\_rand, 'blue')**

**pyplot(x, y, dbscan\_unnormalized\_rand, 'orange')**

**plt.title("DBSCAN Rand")**

**plt.xlabel('eps')**

**plt.ylabel('min\_samples')**

**pyplot(x, y, dbscan\_normalized\_fowlkes, 'blue')**

**pyplot(x, y, dbscan\_unnormalized\_fowlkes, 'orange')**

**plt.title("DBSCAN Fowlkes")**

**plt.xlabel('eps')**

**plt.ylabel('min\_samples')**

***# разные параметры окресности eps\_list***

**eps\_list = np.linspace(2, 3, 10)**

**dbscan\_eps\_unnormalized\_rand = []**

**dbscan\_eps\_normalized\_rand = []**

**dbscan\_eps\_unnormalized\_fowlkes = []**

**dbscan\_eps\_normalized\_fowlkes = []**

***# создание и обучение модели на стандартизированых и исходных данных***

**for eps in eps\_list:**

**dbscan = DBSCAN(eps=eps)**

**predictions = dbscan.fit\_predict(X)**

**rand = rand\_score(california.target[:5000], predictions)**

**dbscan\_eps\_unnormalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_eps\_unnormalized\_fowlkes.append(accuracy\_fowlkes)**

**dbscan = DBSCAN(eps=eps)**

**predictions = dbscan.fit\_predict(X\_scaled)**

**rand = rand\_score(california.target[:5000], predictions)**

**dbscan\_eps\_normalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_eps\_normalized\_fowlkes.append(accuracy\_fowlkes)**

**plot\_metrics(eps\_list, dbscan\_eps\_unnormalized\_rand, dbscan\_eps\_normalized\_rand, 'Залежність точності від значення eps для California Housing Індекс Ранда', 'eps', True)**

**plot\_metrics(eps\_list, dbscan\_eps\_unnormalized\_fowlkes, dbscan\_eps\_normalized\_fowlkes, 'Залежність точності від значення eps для California Housing Індекс Фолкеса-Меллоуза', 'eps', True)**

***# минимальное количество точек, необходимых для формирования плотного кластера***

**min\_samples\_range = range(6, 16)**

**dbscan\_samples\_unnormalized\_rand = []**

**dbscan\_samples\_normalized\_rand = []**

**dbscan\_samples\_unnormalized\_fowlkes = []**

**dbscan\_samples\_normalized\_fowlkes = []**

***# создание и обучение модели на стандартизированых и исходных данных***

**for sample in min\_samples\_range:**

**dbscan = DBSCAN(min\_samples=sample)**

**predictions = dbscan.fit\_predict(X)**

**rand = rand\_score(california.target[:5000], predictions)**

**dbscan\_samples\_unnormalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_samples\_unnormalized\_fowlkes.append(accuracy\_fowlkes)**

**dbscan = DBSCAN(min\_samples=sample)**

**predictions = dbscan.fit\_predict(X\_scaled)**

**rand = rand\_score(california.target[:5000], predictions)**

**dbscan\_samples\_normalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**dbscan\_samples\_normalized\_fowlkes.append(accuracy\_fowlkes)**

**plot\_metrics(min\_samples\_range, dbscan\_samples\_unnormalized\_rand, dbscan\_samples\_normalized\_rand, 'Залежність точності від значення min\_samples для California Housing Індекс Ранда', 'min\_samples', True)**

**plot\_metrics(min\_samples\_range, dbscan\_samples\_unnormalized\_fowlkes, dbscan\_samples\_normalized\_fowlkes, 'Залежність точності від значення min\_samples для California Housing Індекс Фолкеса-Меллоуза', 'min\_samples', True)**

***# контролирует степень изменения доступных кластеров с каждой итерацией***

**damping\_range = np.linspace(0.5, 0.9, 5)**

**ap\_unnormalized\_rand = []**

**ap\_normalized\_rand = []**

**ap\_unnormalized\_fowlkes = []**

**ap\_normalized\_fowlkes = []**

**for damping in damping\_range:**

**apc = AffinityPropagation(damping=damping)**

**predictions = apc.fit\_predict(X)**

**rand = rand\_score(california.target[:5000], predictions)**

**ap\_unnormalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**ap\_unnormalized\_fowlkes.append(accuracy\_fowlkes)**

**apc = AffinityPropagation(damping=damping)**

**predictions = apc.fit\_predict(X\_scaled)**

**rand = rand\_score(california.target[:5000], predictions)**

**ap\_normalized\_rand.append(rand)**

**accuracy\_fowlkes = fowlkes\_mallows\_score(california.target[:5000], predictions)**

**ap\_normalized\_fowlkes.append(accuracy\_fowlkes)**

***# %%***

**plot\_metrics(damping\_range, ap\_unnormalized\_rand, ap\_normalized\_rand, 'Залежність точності від значення damping для California Housing Індекс Ранда')**

***# %%***

**plot\_metrics(damping\_range, ap\_unnormalized\_fowlkes, ap\_normalized\_fowlkes, 'Залежність точності від значення damping для California Housing Індекс Фолкеса-Меллоуза')**

***# %% [markdown]***

***# #### 7) Построение сравнительной диаграмы алгоритмов кластеризации в зависимости от их максимальных значений на точности для ненормализованных и стандартизированых данных.***

***# %%***

***# Находим максимальные значения для KMeans***

**max\_kmeans\_normalized\_rand = np.max(kmeans\_accuracy\_list\_normalized\_rand)**

**max\_kmeans\_unnormalized\_rand = np.max(kmeans\_accuracy\_list\_unnormalized\_rand)**

**max\_kmeans\_normalized\_fowlkes = np.min(kmeans\_accuracy\_list\_normalized\_fowlkes)**

**max\_kmeans\_unnormalized\_fowlkes = np.min(kmeans\_accuracy\_list\_unnormalized\_fowlkes)**

***# Находим максимальные значения для MeanShift***

**max\_meanshift\_normalized\_rand = np.max(meanshift\_accuracy\_normalized\_rand)**

**max\_meanshift\_unnormalized\_rand = np.max(meanshift\_accuracy\_unnormalized\_rand)**

**max\_meanshift\_normalized\_fowlkes = np.min(meanshift\_accuracy\_normalized\_fowlkes)**

**max\_meanshift\_unnormalized\_fowlkes = np.min(meanshift\_accuracy\_normalized\_fowlkes)**

***# Находим максимальные значения для DBSCAN (EPC)***

**max\_dbscan\_eps\_normalized\_rand = np.max(dbscan\_eps\_normalized\_rand)**

**max\_dbscan\_eps\_unnormalized\_rand = np.max(dbscan\_eps\_unnormalized\_rand)**

**max\_dbscan\_eps\_normalized\_fowlkes = np.min(dbscan\_eps\_normalized\_fowlkes)**

**max\_dbscan\_eps\_unnormalized\_fowlkes = np.min(dbscan\_eps\_unnormalized\_fowlkes)**

***# Находим максимальные значения для DBSCAN (min\_sample)***

**max\_dbscan\_sample\_normalized\_rand = np.max(dbscan\_samples\_normalized\_rand)**

**max\_dbscan\_sample\_unnormalized\_rand = np.max(dbscan\_samples\_unnormalized\_rand)**

**max\_dbscan\_sample\_normalized\_fowlkes = np.min(dbscan\_samples\_normalized\_fowlkes)**

**max\_dbscan\_sample\_unnormalized\_fowlkes = np.min(dbscan\_samples\_unnormalized\_fowlkes)**

***# Находим максимальные значения для AffinityPropagation (damping\_range)***

**max\_ap\_normalized\_rand = np.max(ap\_normalized\_rand)**

**max\_ap\_unnormalized\_rand = np.max(ap\_unnormalized\_rand)**

**max\_ap\_normalized\_fowlkes = np.min(ap\_normalized\_fowlkes)**

**max\_ap\_unnormalized\_fowlkes = np.min(ap\_unnormalized\_fowlkes)**

***# %%***

***# вспомогательные массиви для отображения данных на диаграме***

**algorithms = ['KMeans', 'MeanShift', 'DBSCAN (eps)', 'DBSCAN (samples)', 'Affinity Propagation']**

**metrics = ['Rand', 'Fowlkes-Mallows']**

**data\_types = ['Нормалізовані дані', 'Не нормалізовані дані']**

***# %%***

***# словарь для красивого вывода лейблов***

**max\_accuracies = {**

**'KMeans': {**

**'Нормалізовані дані': {**

**'Rand': max\_kmeans\_normalized\_rand,**

**'Fowlkes-Mallows': max\_kmeans\_normalized\_fowlkes**

**},**

**'Не нормалізовані дані': {**

**'Rand': max\_kmeans\_unnormalized\_rand,**

**'Fowlkes-Mallows': max\_kmeans\_unnormalized\_fowlkes**

**}**

**},**

**'MeanShift': {**

**'Нормалізовані дані': {**

**'Rand': max\_meanshift\_normalized\_rand,**

**'Fowlkes-Mallows': max\_meanshift\_normalized\_fowlkes**

**},**

**'Не нормалізовані дані': {**

**'Rand': max\_meanshift\_unnormalized\_rand,**

**'Fowlkes-Mallows': max\_meanshift\_unnormalized\_fowlkes**

**}**

**},**

**'DBSCAN (eps)': {**

**'Нормалізовані дані': {**

**'Rand': max\_dbscan\_eps\_normalized\_rand,**

**'Fowlkes-Mallows': max\_dbscan\_eps\_normalized\_fowlkes**

**},**

**'Не нормалізовані дані': {**

**'Rand': max\_dbscan\_eps\_unnormalized\_rand,**

**'Fowlkes-Mallows': max\_dbscan\_eps\_unnormalized\_fowlkes**

**}**

**},**

**'DBSCAN (samples)': {**

**'Нормалізовані дані': {**

**'Rand': max\_dbscan\_sample\_normalized\_rand,**

**'Fowlkes-Mallows': max\_dbscan\_sample\_normalized\_fowlkes**

**},**

**'Не нормалізовані дані': {**

**'Rand': max\_dbscan\_sample\_unnormalized\_rand,**

**'Fowlkes-Mallows': max\_dbscan\_sample\_unnormalized\_fowlkes**

**}**

**},**

**'Affinity Propagation': {**

**'Нормалізовані дані': {**

**'Rand': max\_ap\_normalized\_rand,**

**'Fowlkes-Mallows': max\_ap\_normalized\_fowlkes**

**},**

**'Не нормалізовані дані': {**

**'Rand': max\_ap\_unnormalized\_rand,**

**'Fowlkes-Mallows': max\_ap\_unnormalized\_fowlkes**

**}**

**}**

**}**

***# %%***

***# вывод диаграммы***

**plt.figure(figsize=(12, 8))**

**for algorithm in algorithms:**

**for data\_type in data\_types:**

**x\_values = [max\_accuracies[algorithm][data\_type][metric] for metric in metrics]**

**plt.scatter(x\_values[0], x\_values[1], label=f'**{algorithm}**(**{data\_type}**)', marker='o')**

**plt.xlabel(metrics[0])**

**plt.ylabel(metrics[1])**

**plt.title('Порівняння максимальних точностей за двома метриками для різних методів кластеризації')**

**plt.legend()**

**plt.grid(True)**

**plt.tight\_layout()**

**plt.show()**

***# %% [markdown]***

***# #### 6) Вивід точностей:***

***# %%***

***# Вывод результатов точности для DBSCAN eps (индекс Рэнда)***

**for i, j in zip(eps\_list, range(len(dbscan\_eps\_unnormalized\_rand))):**

**print(f'**{i:.5f} **|** {dbscan\_eps\_unnormalized\_rand[j]:.5f} **|** {dbscan\_eps\_normalized\_rand[j]:.5f}**')**

***# %%***

***# Вывод результатов точности для DBSCAN eps (индекс Фолкеса-Меллоуза)***

**for i, j in zip(eps\_list, range(len(dbscan\_eps\_unnormalized\_fowlkes))):**

**print(f'**{i:.5f} **|** {dbscan\_eps\_unnormalized\_fowlkes[j]:.5f} **|** {dbscan\_eps\_normalized\_fowlkes[j]:.5f}**')**

***# %%***

***# Вывод результатов точности для DBSCAN min\_samples (индекс Рэнда)***

**for i, j in zip(min\_samples\_range, range(len(dbscan\_samples\_unnormalized\_rand))):**

**print(f'**{i} **|** {dbscan\_samples\_unnormalized\_rand[j]:.5f} **|** {dbscan\_samples\_normalized\_rand[j]:.5f}**')**

***# %%***

***# Вывод результатов точности для DBSCAN min\_samples (индекс Фолкеса-Меллоуза)***

**for i, j in zip(min\_samples\_range, range(len(dbscan\_samples\_unnormalized\_fowlkes))):**

**print(f'**{i} **|** {dbscan\_samples\_unnormalized\_fowlkes[j]:.5f} **|** {dbscan\_samples\_normalized\_fowlkes[j]:.5f}**')**

***# %%***

***# Вывод результатов точности для Affinity Propagation damping (индекс Рэнда)***

**for i, j in zip(damping\_range, range(len(ap\_unnormalized\_rand))):**

**print(f'**{i:.1f} **|** {ap\_unnormalized\_rand[j]:.5f} **|** {ap\_normalized\_rand[j]:.5f}**')**

***# %%***

***# Вывод результатов точности для Affinity Propagation damping (индекс Фолкеса-Меллоуза)***

**for i, j in zip(damping\_range, range(len(ap\_unnormalized\_fowlkes))):**

**print(f'**{i:.1f} **|** {ap\_unnormalized\_fowlkes[j]:.5f} **|** {ap\_normalized\_fowlkes[j]:.5f}**')**