

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 алгоритмів кластеризації щільнісного та графового типу.

Завдання на роботу: завантаження набору даних, формування вхідної вибірки даних, кластеризація із застосуванням алгоритмів DBSCAN та Affinity Propagation, порівняльний аналіз алгоритмів кластеризації.

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

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains three lines of Python code:

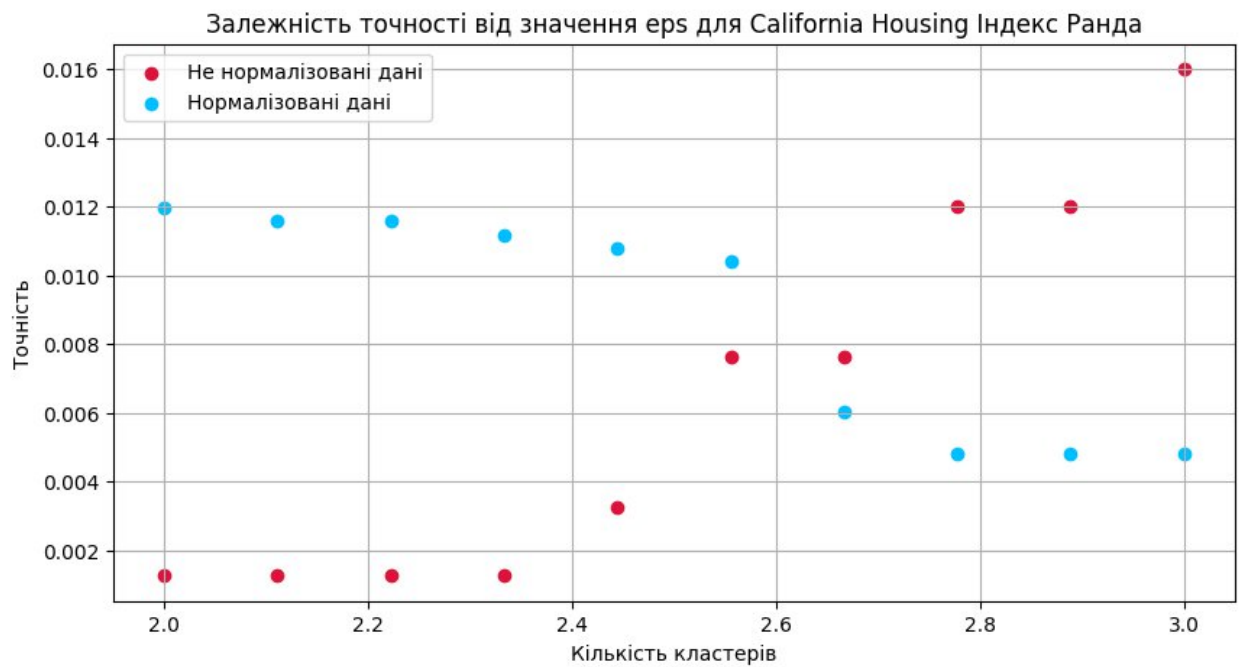
```
1 california = fetch_california_housing()
2 X = california.data[:5000]
3 X_scaled = StandardScaler().fit_transform(X)
```

```
1  california = fetch_california_housing()
2  X = california.data[:5000]
3  X_scaled = StandardScaler().fit_transform(X)
```

Кластеризація методом DBSCAN в залежності від параметру ϵ

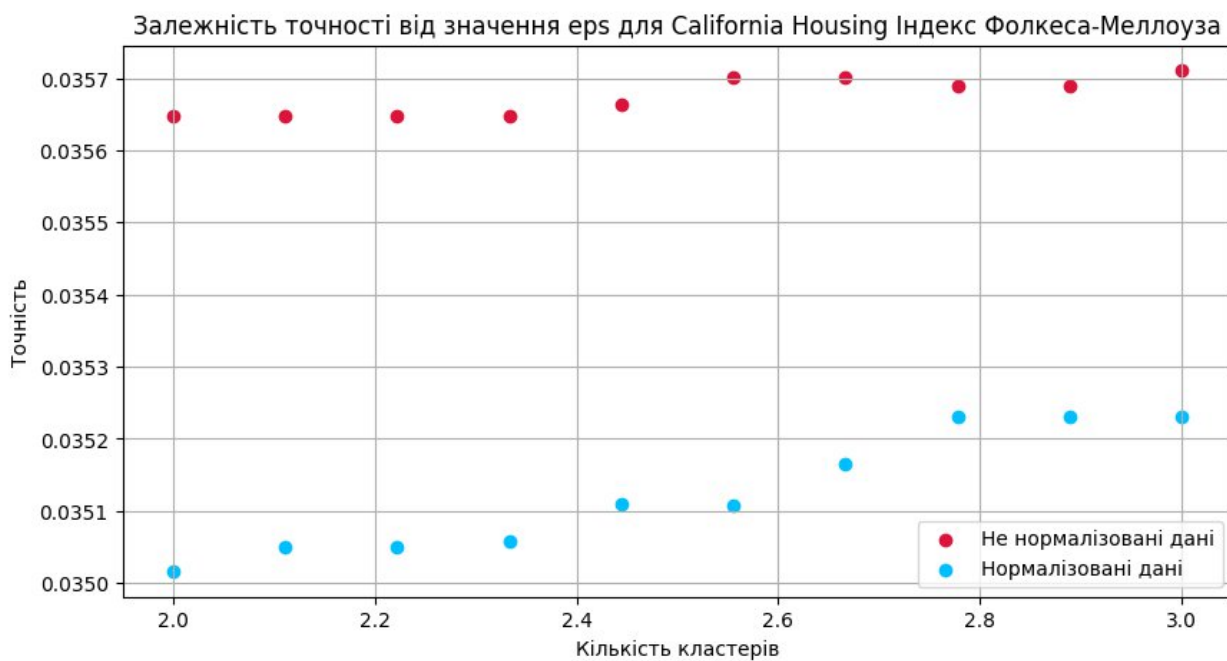
*Залежність точності від значення параметру ϵ в алгоритмі DBSCAN.
Метрика – Індекс Ранда.*

Значення ϵ	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 ϵ в алгоритмі DBSCAN.
Метрика – Індекс Фолкеса-Меллоуза.*

Значення ϵ	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

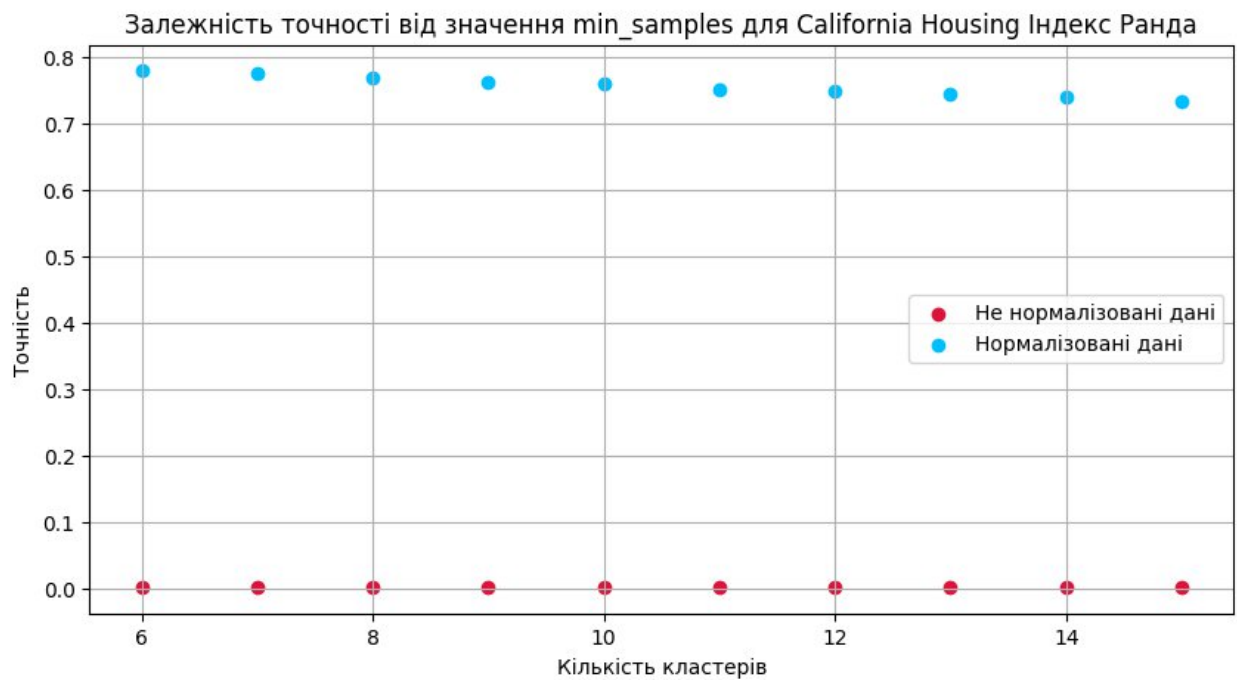


Кластеризація методом DBSCAN в залежності від параметру *min_samples*

*Залежність точності від значення параметру *min_samples* в алгоритм DBSCAN.*

Метрика – Індекс Ранда.

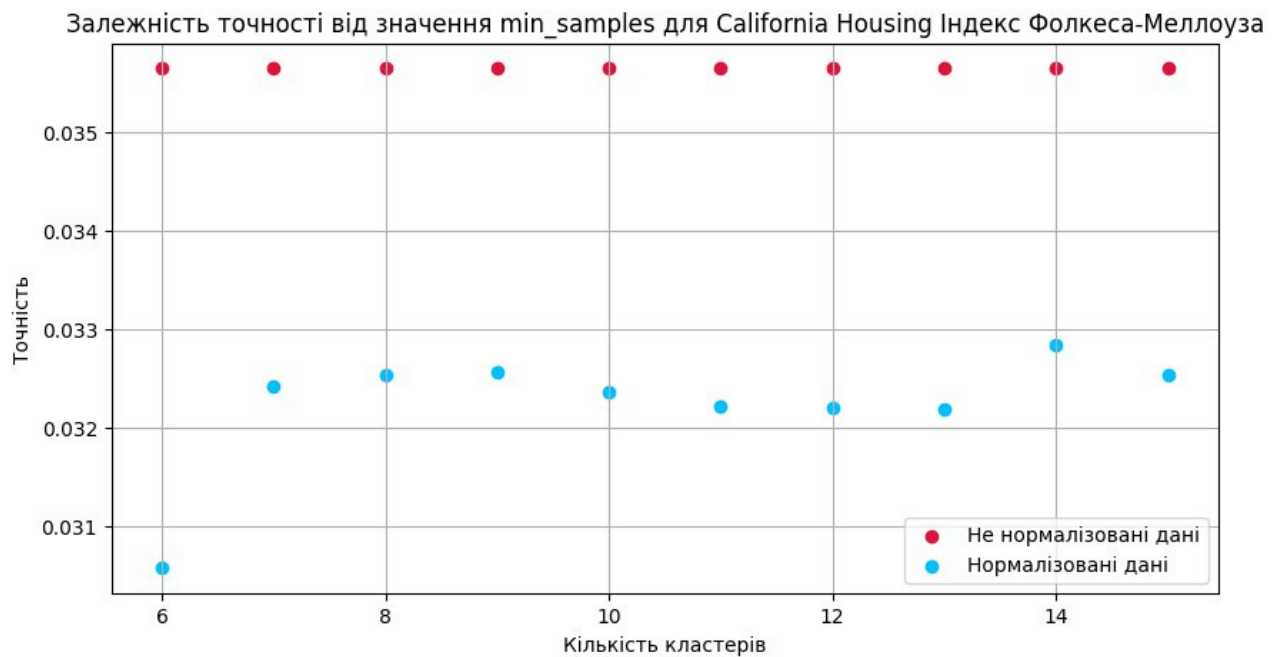
Значення <i>min_samples</i>	Точність на ненормалізованих даних	Точність на нормалізованих даних
6	0.00127	0.77970
7	0.00127	0.77475
8	0.00127	0.76906
9	0.00127	0.76230
10	0.00127	0.75976
11	0.00127	0.75198
12	0.00127	0.74896
13	0.00127	0.74376
14	0.00127	0.74032
15	0.00127	0.73355



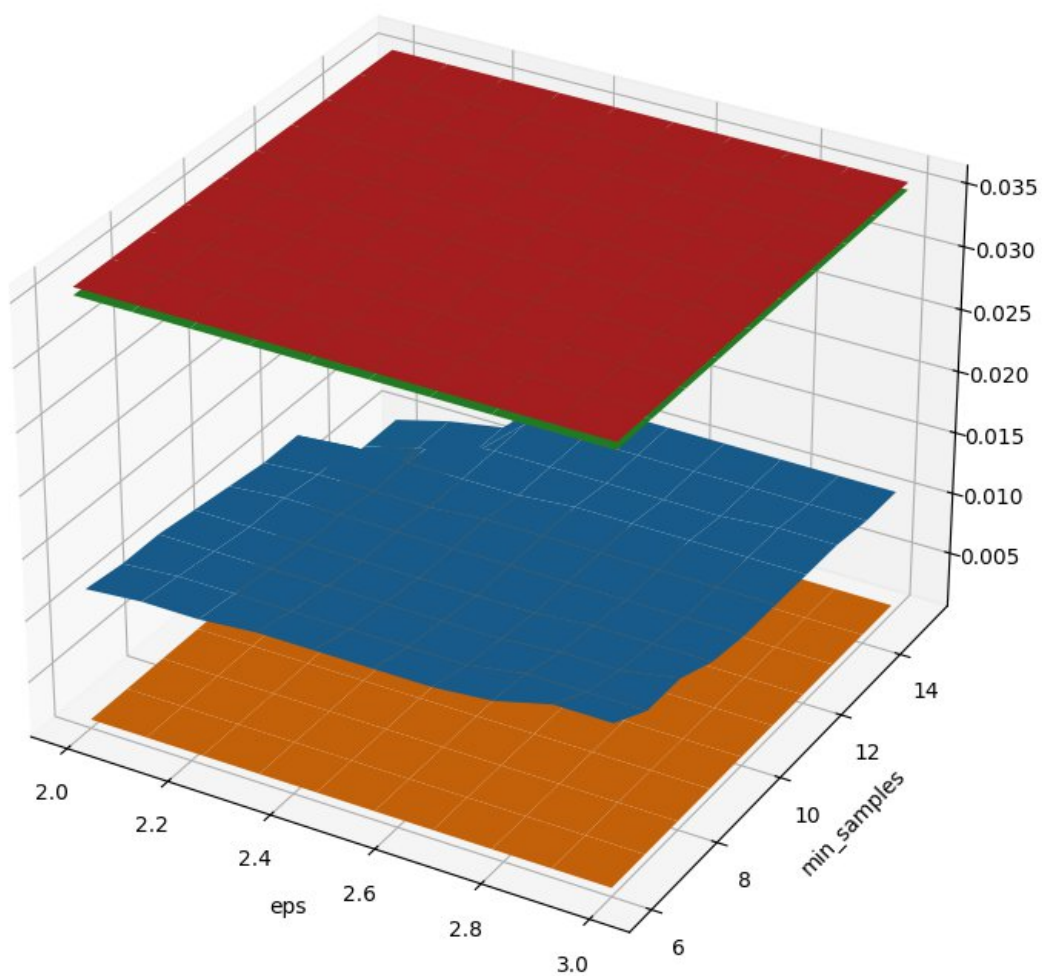
Залежність точності від значення параметру min_samples в алгоритмі DBSCAN.

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*

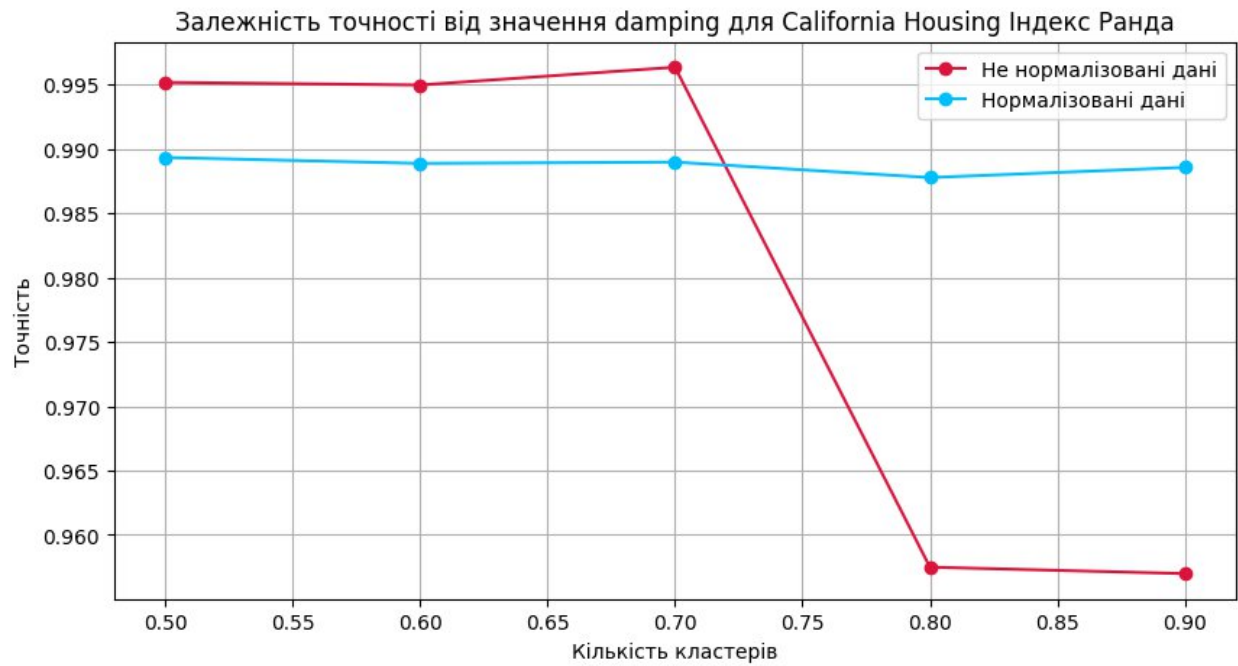


Кластеризація методом Affinity Propagation в залежності від параметру `damping_range`

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

Метрика – Індекс Ранда.

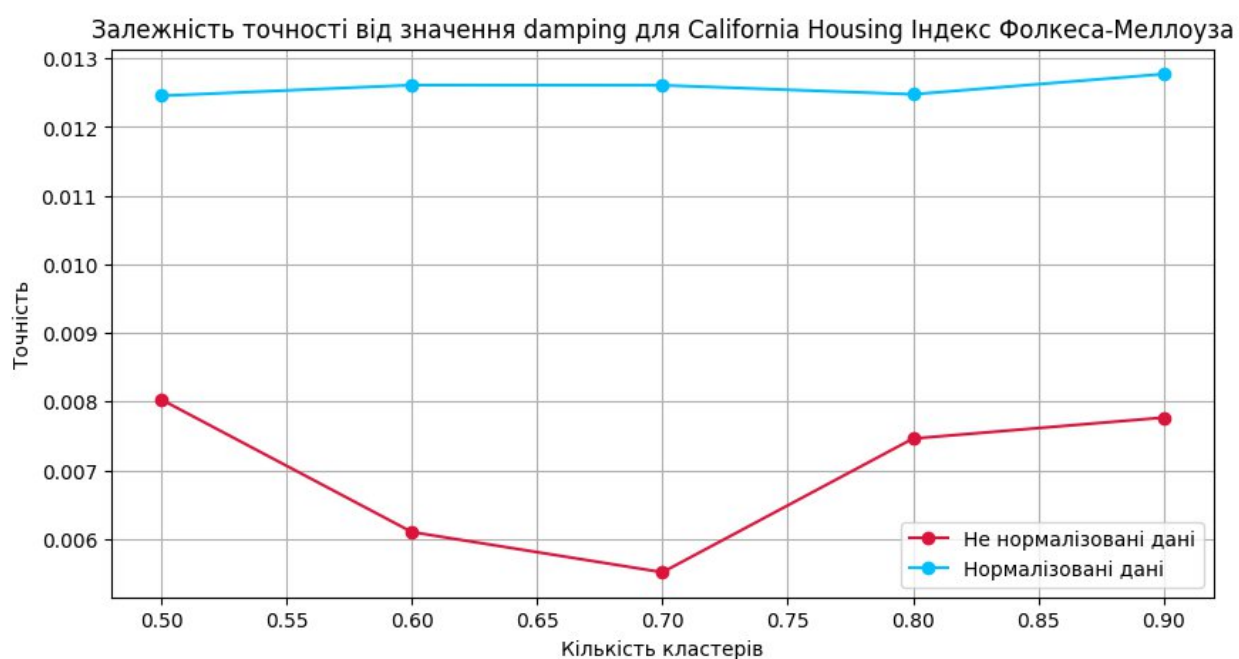
Значення <code>damping</code>	Точність на ненормалізованих даних	Точність на нормалізованих даних
0.5	0.99518	0.98935
0.6	0.99498	0.98888
0.7	0.99636	0.98899
0.8	0.95749	0.98778
0.9	0.95699	0.98857



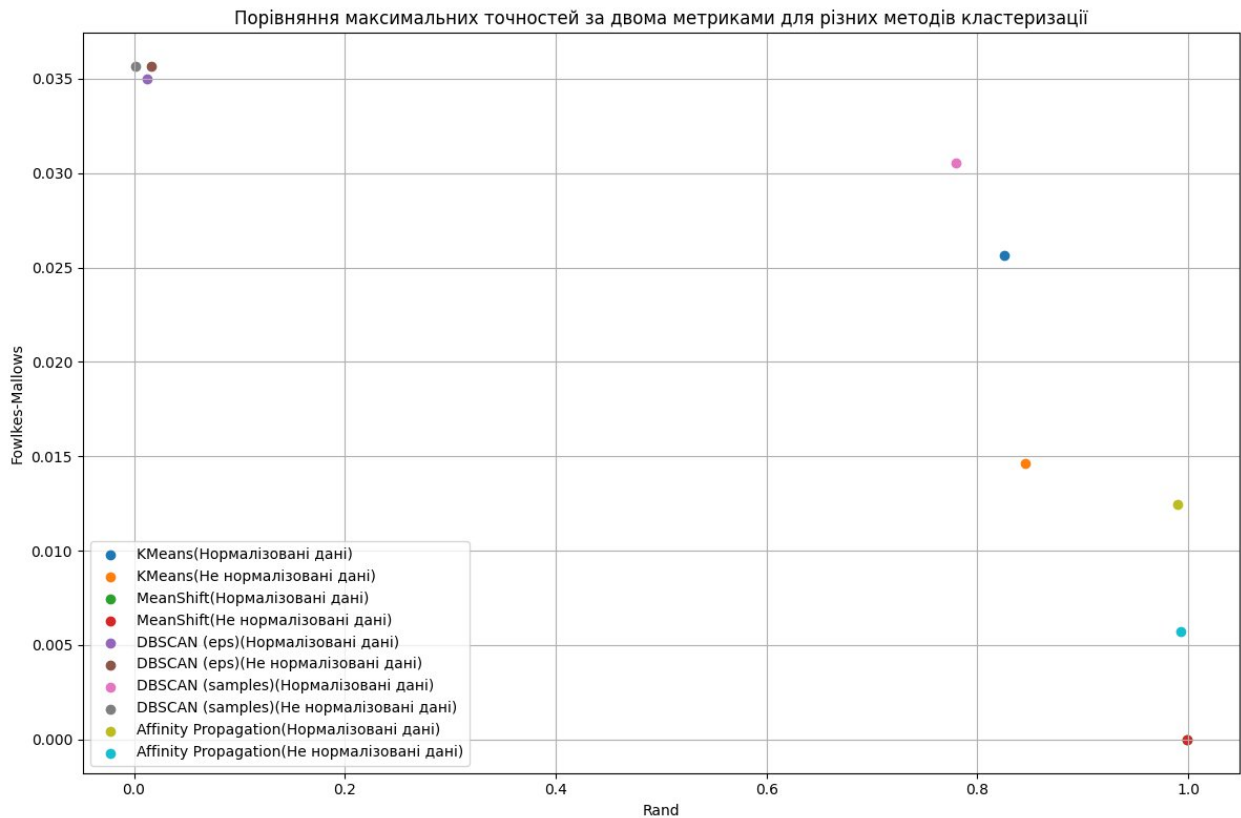
Залежність точності від значення параметру damping в алгоритмі Affinity Propagation.

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



Порівняльна діаграма алгоритмів кластеризації



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)``: Вывести точковый график.

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)

```

параметры 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 окрестности 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
Normalized 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 окрестности 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 стандартизированных 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 стандартизированных 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)
```

```
# контролює ступінь зміни доступних кластерів з кожної
ітерацією
```

```
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 диаграммы алгоритмов кластеризации в
зависимости от их максимальных значений на точности для
ненормализованных и стандартизированных 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_unnormalized_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 массивы для отображения данных на диаграмме
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
        }
    }
}

# %%

# вивод діаграми

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}')

```