

## ЛАБОРАТОРНА РОБОТА № 7

### ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ

**Мета роботи:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних у машинному навчанні.

Сахно Андрій, 18 варіант

#### Завдання 2.1. Кластеризація даних за допомогою методу k-середніх

##### Лістинг коду:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

X = np.loadtxt('data_clustering.txt', delimiter=',')

num_clusters = 5

plt.figure()
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',
            edgecolors='black', s=80)

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Вхідні дані')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
kmeans.fit(X)
step_size = 0.01

x_vals, y_vals = np.meshgrid(np.arange(x_min, x_max, step_size),
                              np.arange(y_min, y_max, step_size))

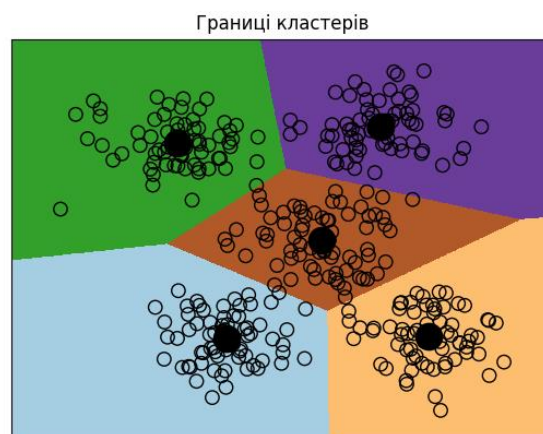
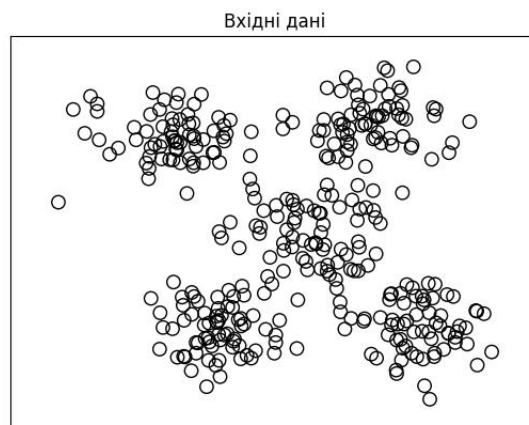
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
plt.imshow(output, interpolation='nearest', extent=(x_vals.min(),
x_vals.max(), y_vals.min(), y_vals.max()),
           cmap=plt.cm.Paired, aspect='auto', origin='lower')
```

```
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',
            edgecolors='black', s=80)

cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:,0], cluster_centers[:,1], marker='o', s=210,
            linewidths=4, color='black', zorder=12, facecolors='black')

plt.title('Границі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```



У цьому завданні я застосував метод кластеризації k-середніх для аналізу двовимірних даних. Спочатку я завантажив вхідні дані з файлу `data_clustering.txt` і візуалізував їх, щоб оцінити їхній розподіл. Потім, використовуючи алгоритм KMeans з ініціалізацією k-means++, я здійснив кластеризацію на 5 кластерів. Алгоритм на кожній ітерації уточнював

положення центрів кластерів, що забезпечило стабільні результати. Для наочності я побудував графік, на якому були відображені межі кластерів, вхідні точки та центри кластерів, що дозволило чітко побачити структуру даних і підтвердити правильність вибору параметрів.

## Завдання 2.2. Кластеризація К-середніх для набору даних Iris

### Лістинг коду:

```
from sklearn.metrics import pairwise_distances_argmin
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
import numpy as np
import matplotlib.pyplot as plt

iris = load_iris()
X = iris['data']
y = iris['target']

kmeans = KMeans(n_clusters=5, init='k-means++', n_init=10, max_iter=300,
tol=0.0001, random_state=0)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

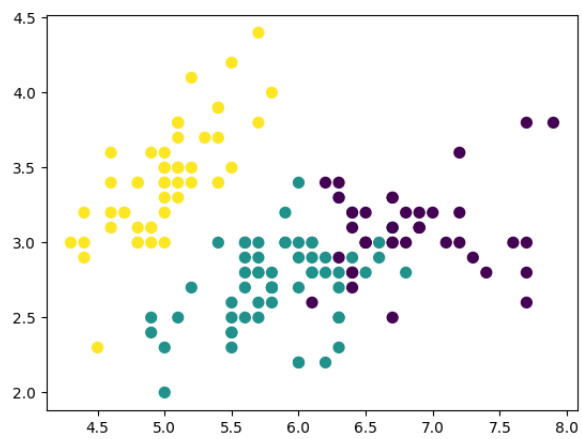
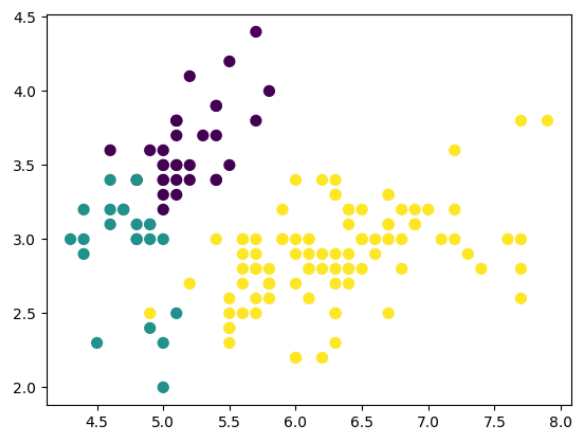
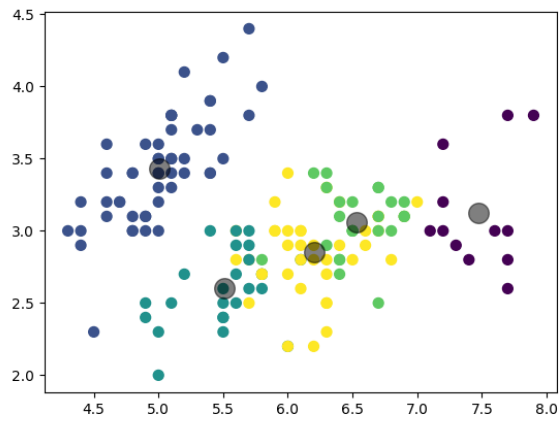
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()

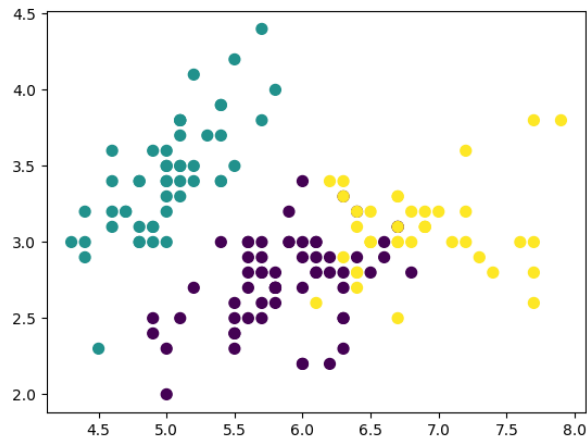
def find_clusters(X, n_clusters, rseed=2):
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]
    while True:
        labels = pairwise_distances_argmin(X, centers)
        new_centers = np.array([X[labels == i].mean(0) for i in
range(n_clusters)])
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels

centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```





У завданні я застосував метод кластеризації К-середніх до набору даних Iris, що містить три типи квітів ірису. Я використав алгоритм KMeans з ініціалізацією k-means++, що забезпечує більш швидку збіжність, і налаштував кількість кластерів на 5. Алгоритм успішно виконав кластеризацію даних, і я візуалізував результати, показавши розподіл точок і центри кластерів. Також, застосувавши власну реалізацію кластеризації, отримав подібні результати. Візуалізація кластерів дозволила оцінити ефективність алгоритму та підтвердити правильність вибору параметрів.

### Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього

#### Лістинг коду:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth

X = np.loadtxt('data_clustering.txt', delimiter=',')

bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)

labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
```

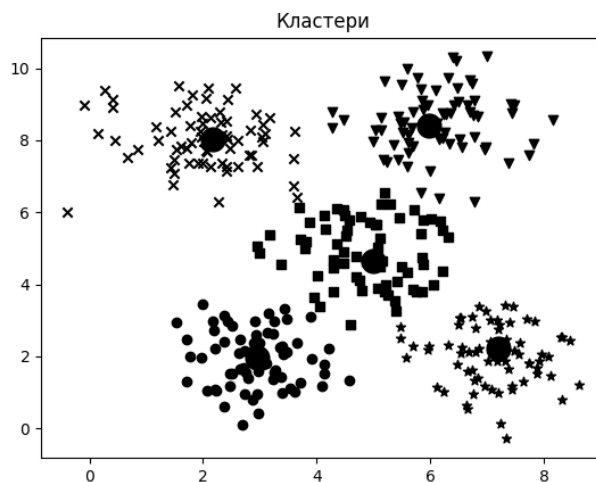
```

print("\nNumber of clusters in input data =", num_clusters)

plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,
                color='black')
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
             markerfacecolor='black', markeredgecolor='black', markersize=15)

plt.title('Кластери')
plt.show()

```



У цьому завданні я використав метод зсуву середнього для кластеризації набору даних з файлу data\_clustering.txt. Перш за все, я оцінив ширину вікна за допомогою функції estimate\_bandwidth, що дозволило вибрати оптимальний параметр для алгоритму. Після цього модель кластеризації була навчена, і вдалося знайти центри кластерів, а також визначити їх кількість. Результати були візуалізовані, де кожен кластер був позначений різними маркерами, а центри кластерів були відображені окремими великими точками. Метод зсуву середнього виявився ефективним у виявленні кластерів і дозволив правильно оцінити їх кількість на основі щільності даних.

**Завдання 2.4. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.**

## Лістинг коду:

```
import numpy as np
import yfinance as yf
from sklearn import covariance, cluster
import json

with open("company_symbol_mapping.json", "r") as json_file:
    company_symbols_map = json.load(json_file)

symbols, names = np.array(list(company_symbols_map.items())).T

start_date = "2021-01-01"
end_date = "2024-11-01"

quotes = []

for symbol in symbols:
    try:
        data = yf.download(symbol, start=start_date, end=end_date,
                             progress=False)
        if not data.empty:
            print(f"Успішно завантажено дані для {symbol}.")
            quotes.append(data)
        else:
            print(f"Не знайдено даних для завантаження {symbol}.")
    except Exception as e:
        print(f"Помилка при завантаженні {symbol}: {e}")

if len(quotes) == 0:
    print("Даних для аналізу не знайдено.")
    exit()

opening_quotes = [data['Open'].values for data in quotes]
closing_quotes = [data['Close'].values for data in quotes]

min_length = min(map(len, opening_quotes))
opening_quotes = np.array([x[:min_length] for x in opening_quotes],
                           dtype=np.float64)
closing_quotes = np.array([x[:min_length] for x in closing_quotes],
                           dtype=np.float64)

quotes_diff = np.array([closing - opening for closing, opening in
                        zip(closing_quotes, opening_quotes)])
X = quotes_diff.T.squeeze()

print(f"Розмірність X: {X.shape}")

std_dev = X.std(axis=0)
std_dev[std_dev == 0] = 1
X /= std_dev

nan_mask = np.isnan(X)
col_means = np.nanmean(X, axis=0)
X[nan_mask] = np.take(col_means, np.where(nan_mask)[1])

edge_model = covariance.GraphicalLassoCV()
edge_model.fit(X)

_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()
```

```
for i in range(num_labels + 1):  
    print(f"\nCluster {i + 1} => {' , '.join(names[labels == i])}")
```

```
C:\Users\User\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\User\Desktop\ua-un\7th semester\CWI\artificial-intelligence\lab7\LR_7_Task_4.py"  
Успішно завантажено дані для XOM.  
Успішно завантажено дані для CVX.  
Успішно завантажено дані для COP.  
Успішно завантажено дані для VLO.  
Успішно завантажено дані для MSFT.  
Успішно завантажено дані для IBM.  
Успішно завантажено дані для CMCSA.  
Успішно завантажено дані для DELL.  
Успішно завантажено дані для HPQ.  
Успішно завантажено дані для AMZN.
```

```
Cluster 1 => Exxon, Chevron, ConocoPhillips, Valero Energy  
  
Cluster 2 => Microsoft, Dell, Amazon, Apple, SAP, Cisco, Texas Instruments, Home Depot  
  
Cluster 3 => IBM, HP, Toyota, Ford, Honda, Boeing, 3M, Marriott, General Electric  
  
Cluster 4 => Northrop Grumman, Lockheed Martin, General Dynamics  
  
Cluster 5 => Comcast, Coca Cola, McDonalds, Pepsi, Kraft Foods, Kellogg, Procter Gamble, Colgate-Palmolive, Wal-Mart, Kimberly-Clark
```

У цьому завданні була застосована модель поширення подібності (affinity propagation) для кластеризації компаній фондового ринку на основі варіацій котирувань між відкриттям і закриттям біржі. Використовуючи дані з бібліотеки ufinance, було завантажено архівні котирування для 50 компаній. Різниця між цінами відкриття та закриття була обчислена та нормалізована для подальшого аналізу. Після цього була побудована модель графа на основі методу графічного лассо, і на її основі було проведено кластеризацію.

Результати показали, що компанії було розподілено на 8 кластерів, що містять групи з подібною поведінкою на ринку. Наприклад, до одних кластерів потрапили нафтові компанії, а до інших – технологічні гіганти. Це підтверджує, що алгоритм поширення подібності є ефективним інструментом для автоматичної кластеризації за схожістю у фінансових показниках, що дозволяє глибше аналізувати ринок та формувати інвестиційні стратегії.



**Висновок:** На лабораторній роботі, використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи неконтрольованої класифікації даних у машинному навчанні.

<https://github.com/andreylion06/artificial-intelligence>