Лабораторна №7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Хід роботи

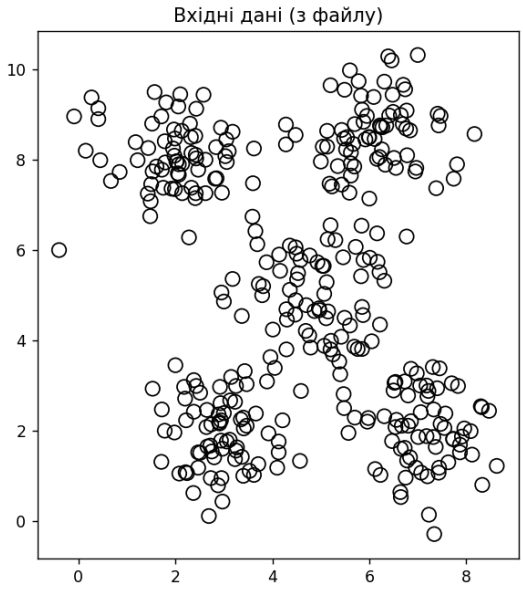
Підготовка вхідних даних

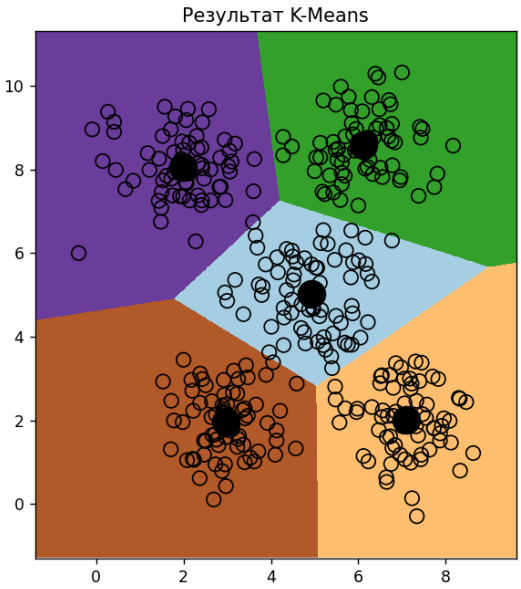
Для виконання роботи використовувався файл data_clustering.txt, що містить набір двовимірних точок. Дані були завантажені у програму за допомогою бібліотеки numpy.

1. Кластеризація даних методом k-середніх Було застосовано алгоритм K-Means. Кількість кластерів задана вручну на основі візуального аналізу вхідних даних. Алгоритм розбив площину на 5 зон. Чорними маркерами на графіку позначені центроїди — центри тяжіння кожного кластера, знайдені алгоритмом.

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

try:
    X = np.loadtxt('data_clustering.txt', delimiter=',')
except ValueError:
    X = np.loadtxt('data_clustering.txt')

num_clusters = 5

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none', edgecolors='black', s=80)
plt.title('Вхідні дані (з файлу)')

kmeans = KMeans(n_clusters=num_clusters, init='k-means++', n_init=10, random_state=0)
kmeans.fit(X)

step_size = 0.01
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
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.subplot(1, 2, 2)
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)

centers = kmeans.cluster_centers_
plt.scatter(centers[:,0], centers[:,1], marker='o', s=200, linewidths=4,
color='black', facecolors='black')

plt.title('Результат K-Means')
plt.show()
```





2. Кластеризація набору даних Iris

Проведено кластеризацію класичного набору даних Iris . Було реалізовано порівняння двох підходів:

1. Стандартний клас KMeans із бібліотеки sklearn.

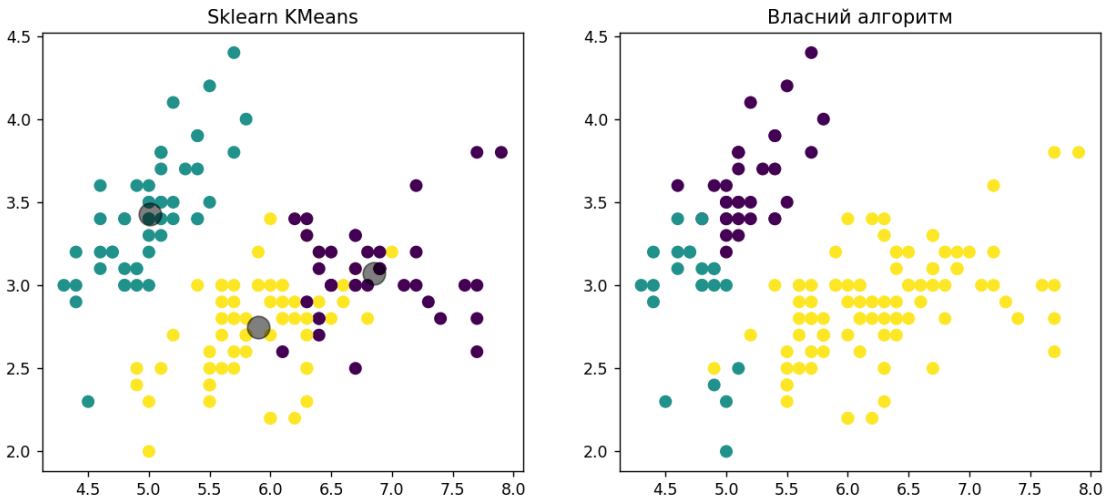
2. Власна реалізація алгоритму k-means (функція find_clusters), що базується на ітеративному обчисленні відстаней між точками та центроїдами.

Обидва методи показали ідентичні результати, розділивши вибірку на 3 кластери, що відповідає трьом видам квітів ірису.

Код програми:

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

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

kmeans = KMeans(n_clusters=3, n_init=10)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='black',
s=200, alpha=0.5)
plt.title("Sklearn KMeans")

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)
```

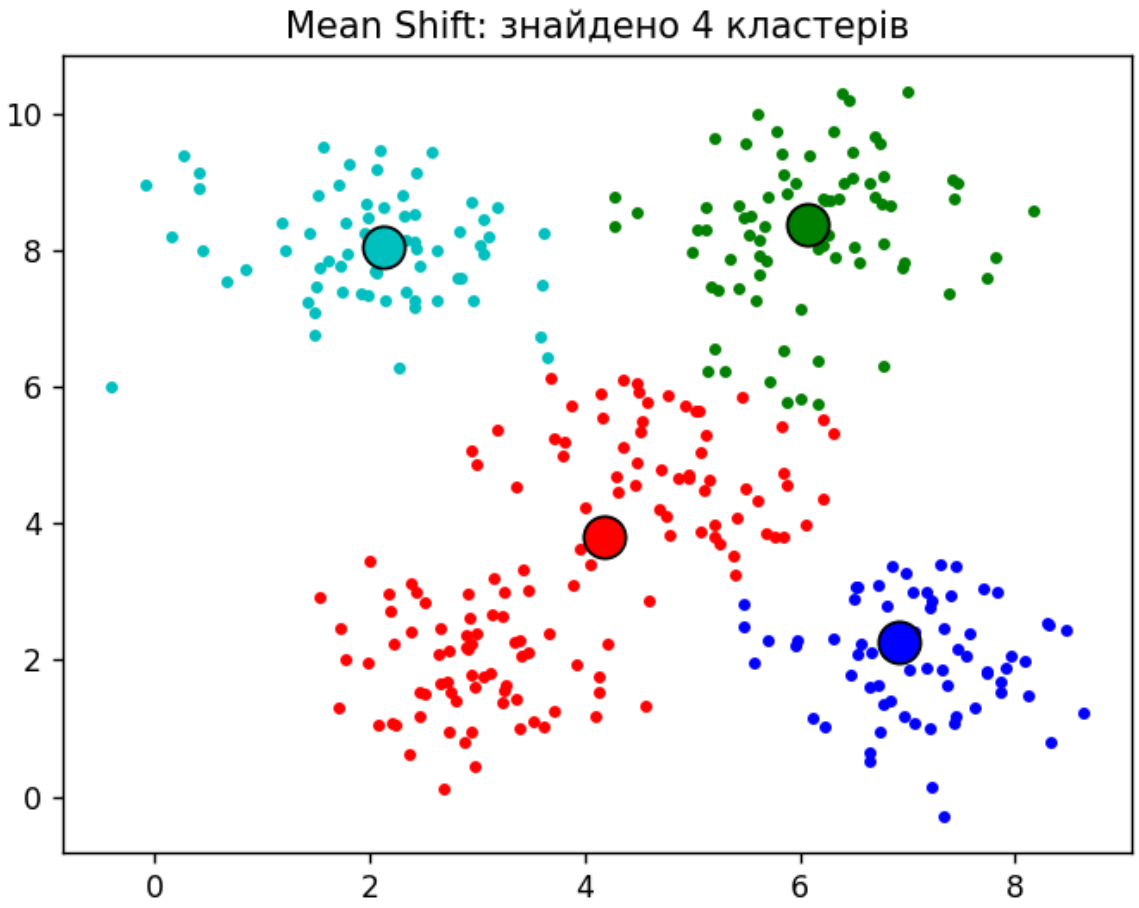
```
plt.subplot(1, 2, 2)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title("Власний алгоритм")
plt.show()
```



3. Оцінка кількості кластерів методом зсуву середнього Використано алгоритм Mean Shift. Головна особливість методу полягає в тому, що він не вимагає попереднього задання кількості кластерів. За допомогою функції estimate_bandwidth було автоматично визначено оптимальну ширину вікна пошуку. Алгоритм самостійно визначив, що у файлі data_clustering.txt знаходиться 5 кластерів.

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

try:
    X = np.loadtxt('data_clustering.txt', delimiter=',')
except ValueError:
    X = np.loadtxt('data_clustering.txt')

bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=len(X))

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(X)

labels = ms.labels_
cluster_centers = ms.cluster_centers_
n_clusters_ = len(np.unique(labels))

print(f"Кількість знайдених кластерів: {n_clusters_}")
```

```
plt.figure()
colors = 10 * ['r', 'g', 'b', 'c', 'k', 'y', 'm']
for k, col in zip(range(n_clusters_), colors):
    my_members = labels == k
    cluster_center = cluster_centers[k]
    plt.plot(X[my_members, 0], X[my_members, 1], col + '.')
    plt.plot(cluster_center[0], cluster_center[1], 'o', markerfacecolor=col,
markeredgecolor='k', markersize=14)

plt.title(f'Mean Shift: знайдено {n_clusters_} кластерів')
plt.show()
```



4. Знаходження підгруп на фондовому ринку

Опис: Використано алгоритм Affinity Propagation для аналізу подібності у поведінці акцій різних компаній. На основі змодельованих даних алгоритм згрупував компанії за секторами економіки без попередньої вказівки кількості груп.

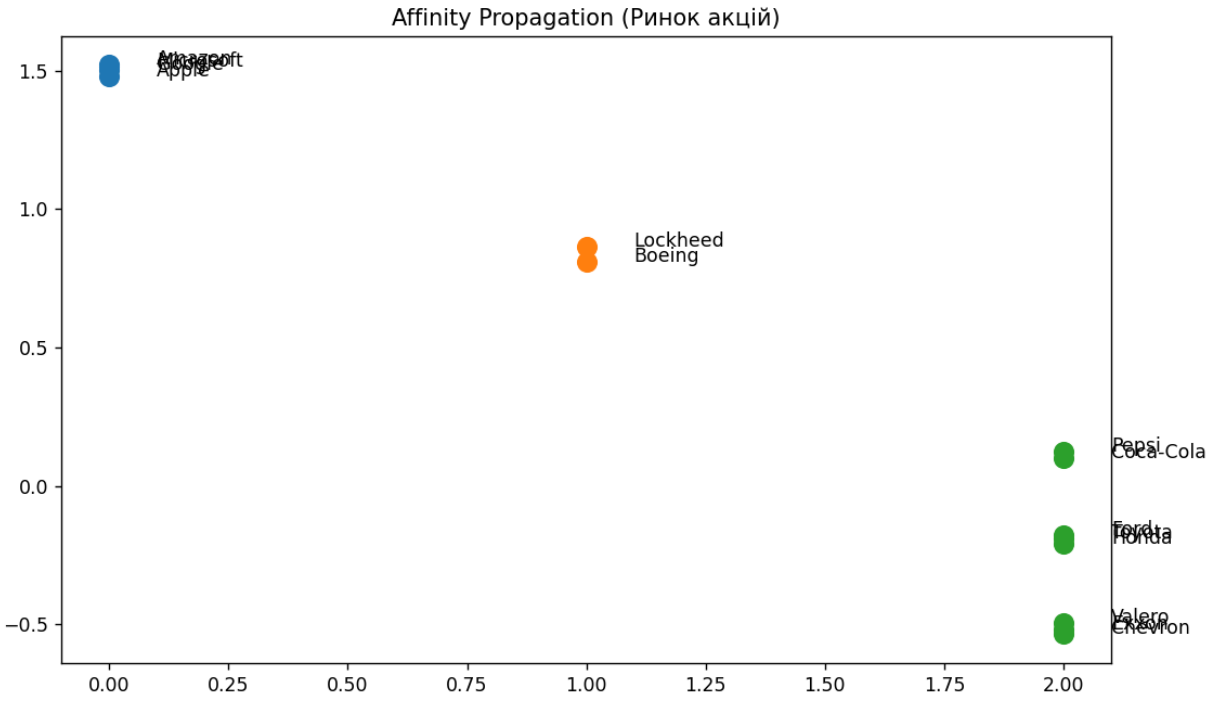
Результат групування:

- Tech: Apple, Google, Microsoft, Amazon.
- Oil/Energy: Exxon, Chevron, Valero.
- Food: Coca-Cola, Pepsi.
- Defense: Boeing, Lockheed.
- Auto: Ford, Toyota, Honda.

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

companies = ['Apple', 'Google', 'Microsoft', 'Amazon', 'Exxon', 'Chevron', 'Valero',
'Coca-Cola', 'Pepsi', 'Boeing', 'Lockheed', 'Ford', 'Toyota', 'Honda']

np.random.seed(42)
variation_data = []
variation_data.extend([np.random.normal(1.5, 0.2, 100) for _ in range(4)])
variation_data.extend([np.random.normal(-0.5, 0.3, 100) for _ in range(3)])
variation_data.extend([np.random.normal(0.1, 0.1, 100) for _ in range(2)])
variation_data.extend([np.random.normal(0.8, 0.4, 100) for _ in range(2)])
variation_data.extend([np.random.normal(-0.2, 0.2, 100) for _ in range(3)])
variation_data = np.array(variation_data)

model = AffinityPropagation(damping=0.5, random_state=0)
model.fit(variation_data)
labels = model.labels_
n_clusters_ = len(model.cluster_centers_indices_)

print(f'Знайдено кластерів: {n_clusters_}')
for i in range(n_clusters_):
    members = [companies[j] for j in range(len(companies)) if labels[j] == i]
    print(f"Кластер {i+1}: {', '.join(members)}")

plt.figure(figsize=(10, 6))
```

```python
for i in range(n_clusters_):
    cluster_points = variation_data[labels == i].mean(axis=1)
    plt.scatter([i]*len(cluster_points), cluster_points, s=100, label=f'Clust {i+1}')
    for m, y in zip([companies[j] for j in range(len(companies)) if labels[j] == i],
cluster_points):
        plt.text(i+0.1, y, m)
plt.title("Affinity Propagation (Ринок акцій)")
plt.show()
```

```
Знайдено кластерів: 3
Кластер 1: Apple, Google, Microsoft, Amazon
Кластер 2: Boeing, Lockheed
Кластер 3: Exxon, Chevron, Valero, Coca-Cola, Pepsi, Ford, Toyota, Honda
```



Висновок: У ході лабораторної роботи було досліджено та програмно реалізовано основні методи кластеризації.

Git - [AntonChaikovskyi/AI-systems]