ЛАБОРАТОРНА РОБОТА № 7

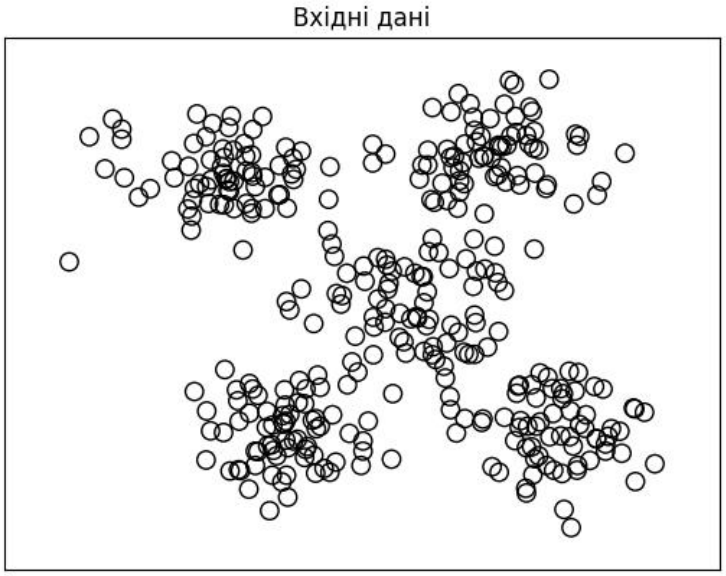
ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ

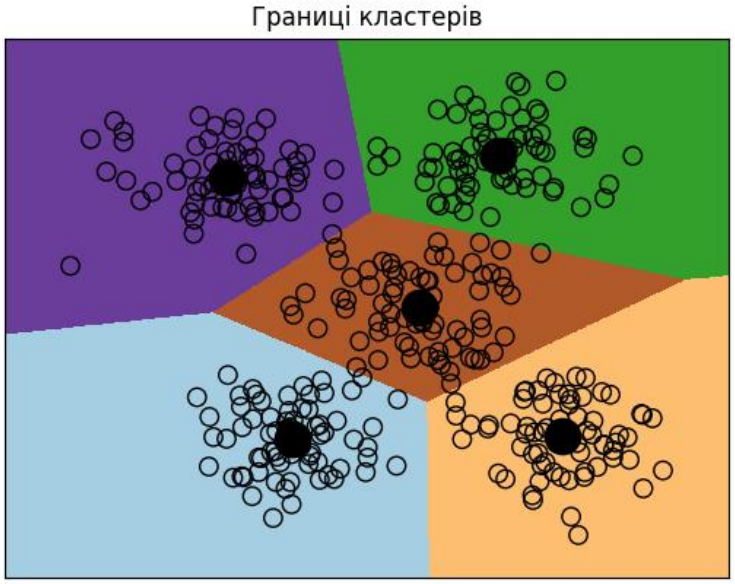
Лукашевич Влада ІПЗ-21-1

https://github.com/vladalukashevych/artificial-intelligence-systems

Завдання №1. Кластеризація даних за допомогою методу k-середніх.

```
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()
```

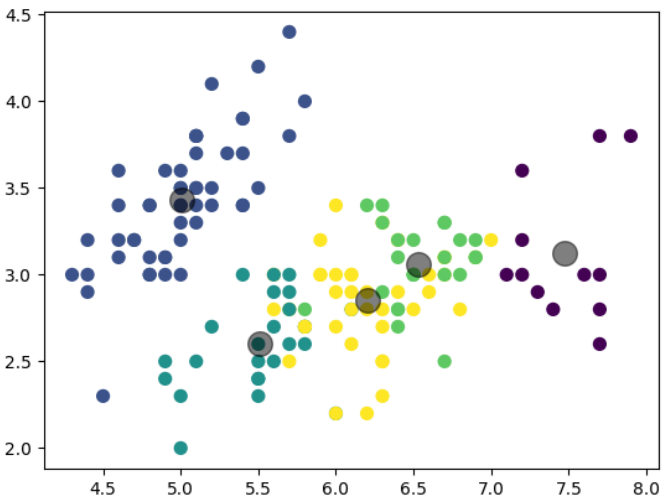


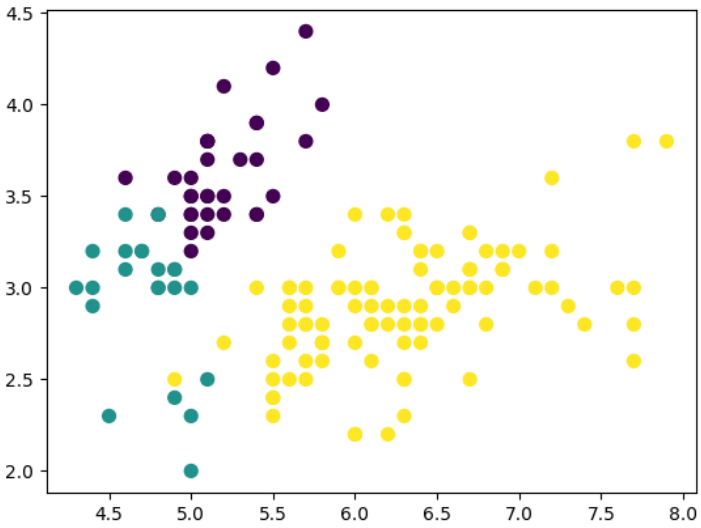


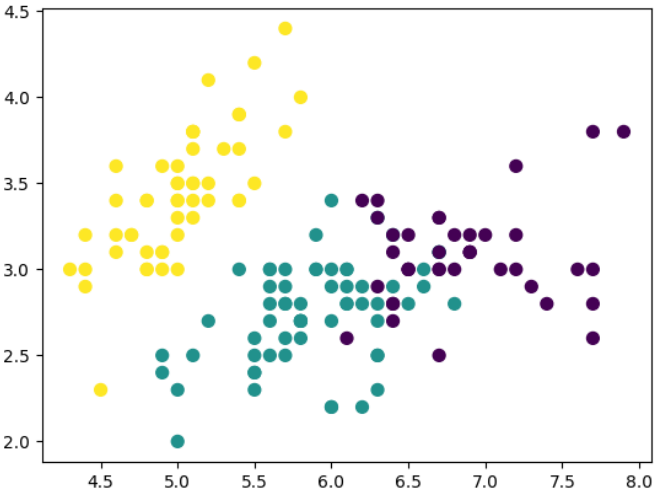
Програма реалізує кластеризацію точок з файлу data_clustering.txt за допомогою методу K-means із використанням ініціалізації k-means++, що підвищує точність алгоритму шляхом уникнення випадкових виборів початкових центрів. Для візуалізації результатів створюється сітка координат, яка дозволяє побудувати межі кластерів та відобразити їхній розподіл на площині.

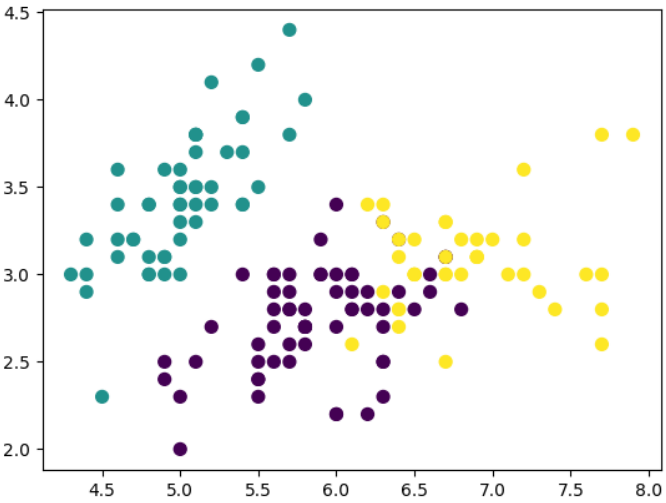
Завдання №2. Кластеризація K-середніх для набору даних Iris.

```
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()
```




4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

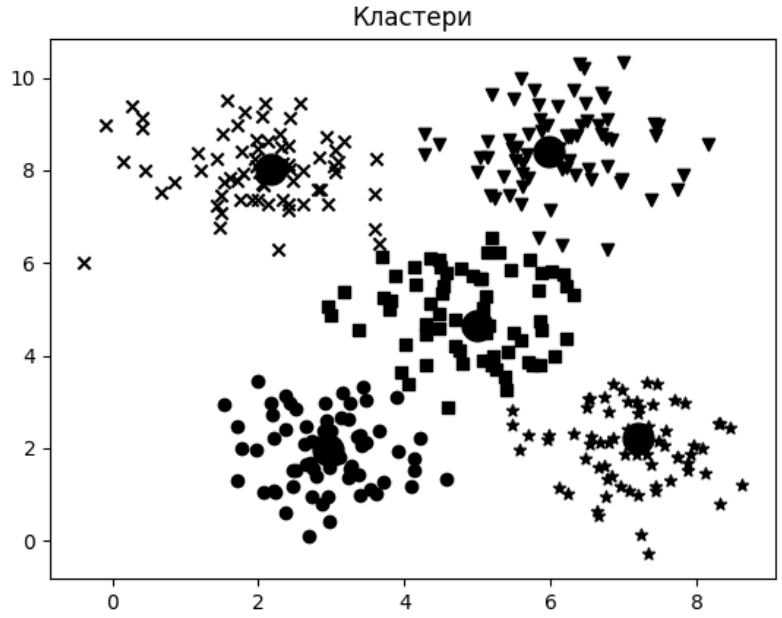

4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

Програма здійснює кластеризацію даних з набору Iris за допомогою методу K-means, орієнтуючись на кількість класів у даних. Реалізовано як стандартний підхід K-means, так і альтернативний метод кластеризації через функцію find_clusters, яка використовує випадковий вибір початкових центроїдів.

Завдання №3. Оцінка кількості кластерів з використанням методу зсуву середнього.

```
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()
```

```
Centers of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Number of clusters in input data = 5
```



Цей код виконує кластеризацію даних за допомогою алгоритму Mean Shift. Спочатку обчислюється параметр bandwidth, необхідний для визначення кластерів, після чого модель тренується на завантажених даних. У результаті програма знаходить центри кластерів і відображає їх на графіку, використовуючи різні маркери для кожного кластеру.

Завдання №4. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.

```
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"Дані успішно завантажено успішно 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}")
```

```python
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"\n\tCluster {i + 1} => {', '.join(names[labels == i])}")
```

```
Дані успішно завантажено успішно для CVS.
Дані успішно завантажено успішно для CAT.
Дані успішно завантажено успішно для DD.
Розмірність X: (964, 50)
```

```
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 Electrics

Cluster 4 => Northrop Grumman, Lookheed Martin, General Dynamics

Cluster 5 => Comcast, Coca Cola, Mc Donalds, Pepsi, Kraft Foods, Kellogg, Procter Gamble, Colgate-Palmolive, Wal-Mart, Kimberly-Clark

Cluster 6 => Wells Fargo, JPMorgan Chase, AIG, American express, Bank of America, Goldman Sachs, Xerox, Ryder, Caterpillar, DuPont de Nemours

Cluster 7 => GlaxoSmithKline, Sanofi-Aventis, Novartis

Cluster 8 => Walgreen, Pfizer, CVS
```

Програма демонструє роботу неконтрольованого навчання на прикладі кластеризації акцій різних компаній. Дані було завантажено за допомогою бібліотеки yfinance, а ознаки сформовані на основі різниці між вартістю відкриття та закриття для кожної акції.

Дані були стандартизовані та очищені від пропущених значень, для коректної подальшої роботи. Була використана методика графових моделей, яка допомогла виявити структуру зв'язків між акціями. На основі отриманої коваріаційної матриці було здійснено кластеризацію з використанням Affinity Propagation.

Висновок: виконуючи лабораторну роботу, я дослідила методи неконтрольованої класифікації даних у машинному навчанні, використовуючи спеціалізовані бібліотеки та мову програмування Python.