ЛАБОРАТОРНА РОБОТА № 7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Варіант 15

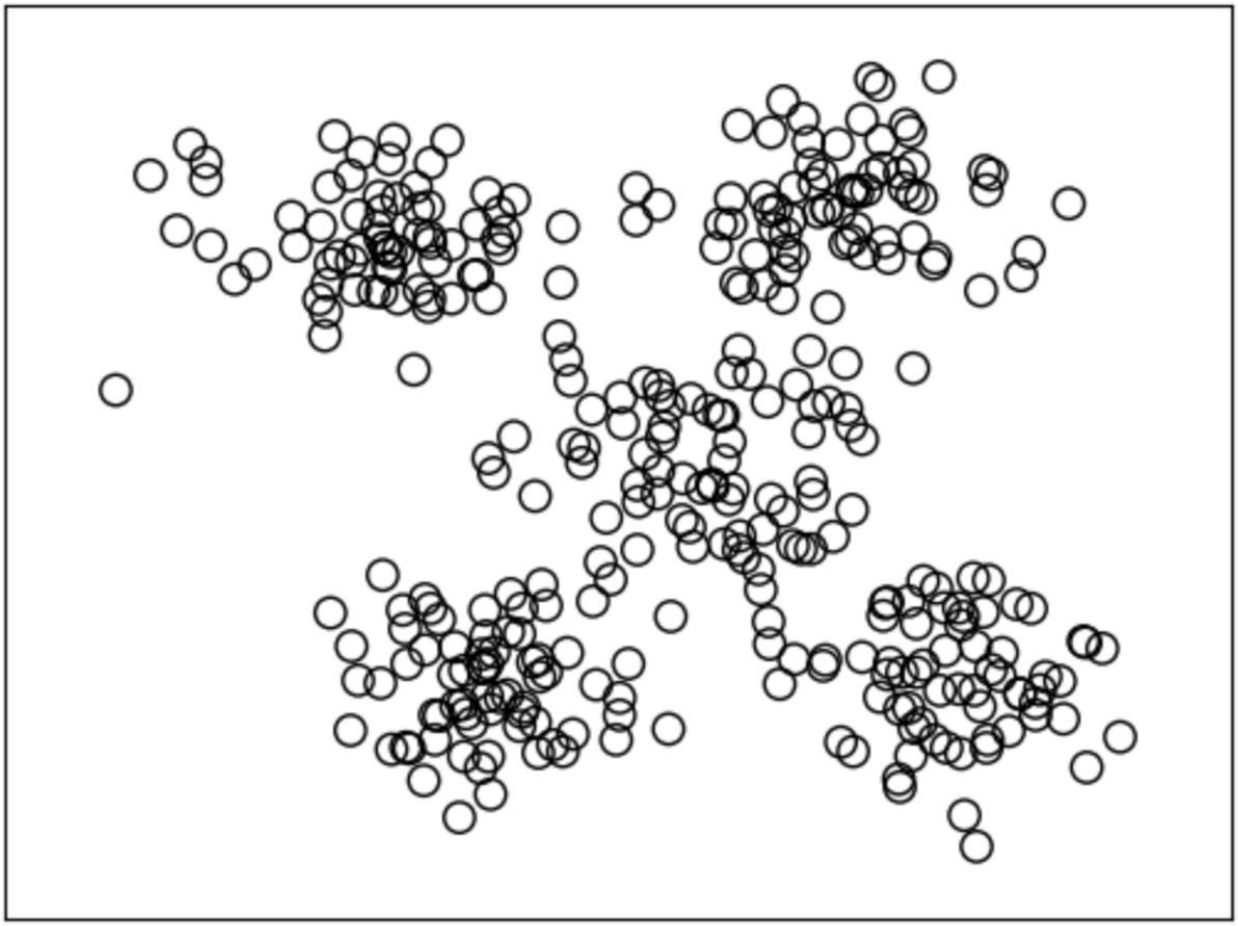
Хід роботи:

Завдання 2.1. Кластеризація даних за допомогою методу k-середніх

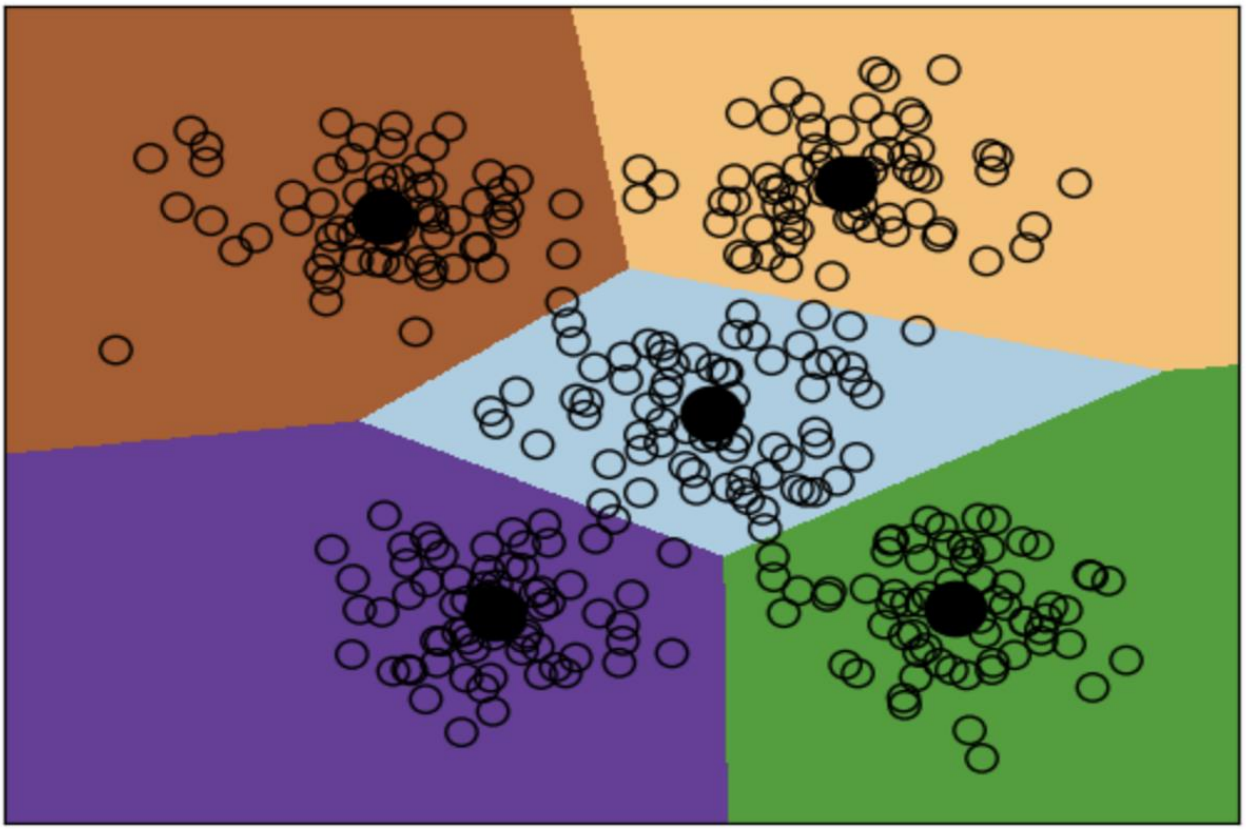
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics

# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')

num_clusters = 5

# Включення вхідних даних до графіка
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 данні')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

plt.show()

# Створення та навчання моделі кластеризації KMeans
kmeans = KMeans( init='k-means++',n_clusters=num_clusters, n_init=10)
kmeans.fit(X)

# Визначення кроку сітки
step_size = 0.01

#Відображення точок сітки
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

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

# Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

# Графічне відображення областей та виділення їх кольором
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'
)

# Відображення вхідних точок
plt.scatter(
    X[:, 0], X[:, 1],
    marker='o',
    facecolors='none',
    edgecolors='black',
    s=80
)
# Відображення центрів кластерів
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'
)
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('Границі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Вхідні данні



Границі кластерів

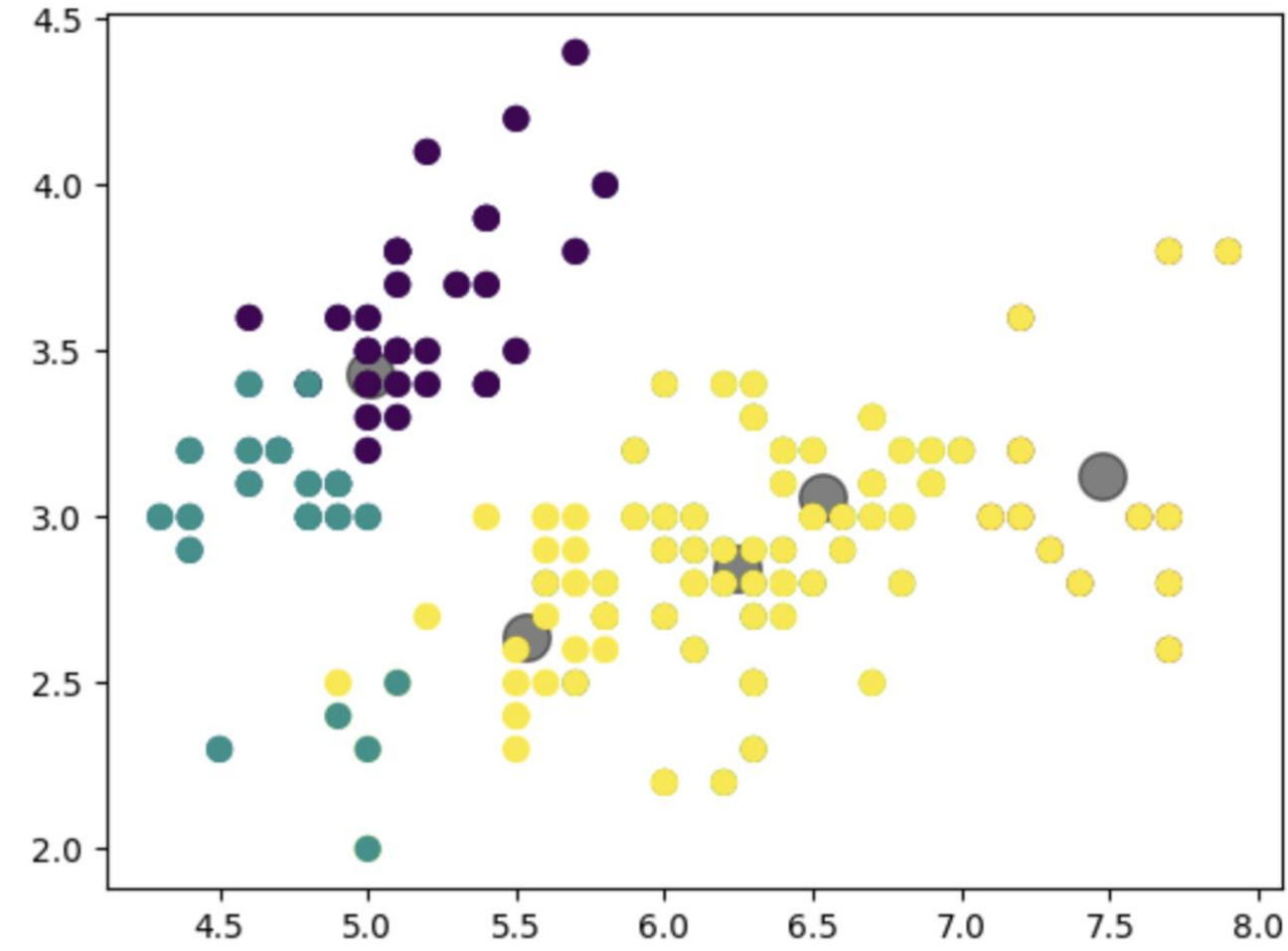


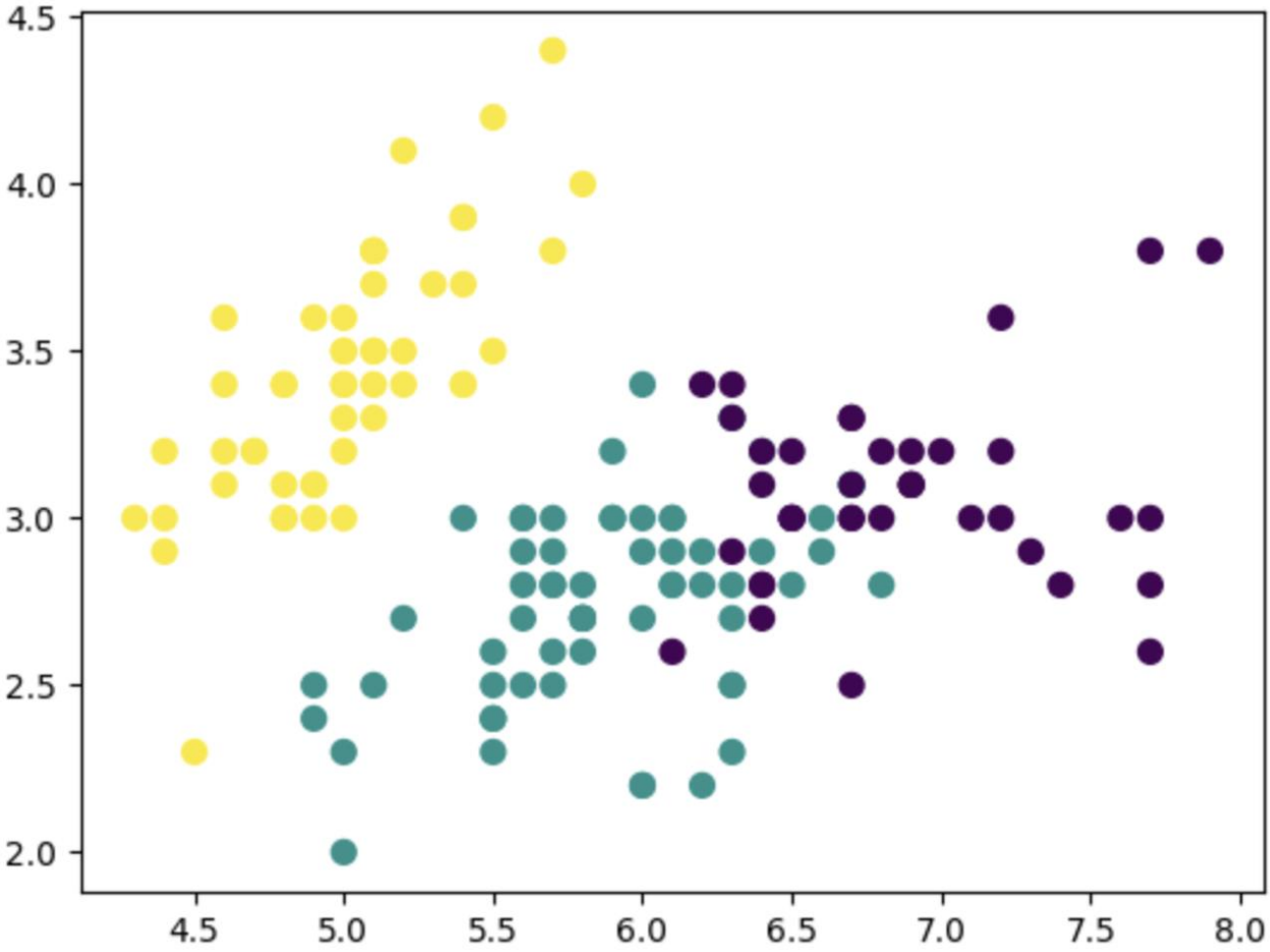
Висновок: Метод k-середніх забезпечив ефективний поділ вибірки на визначену кількість кластерів. Візуалізація результатів чітко показала розташування центрів та меж кластерів, підтверджуючи якість проведеного аналізу.

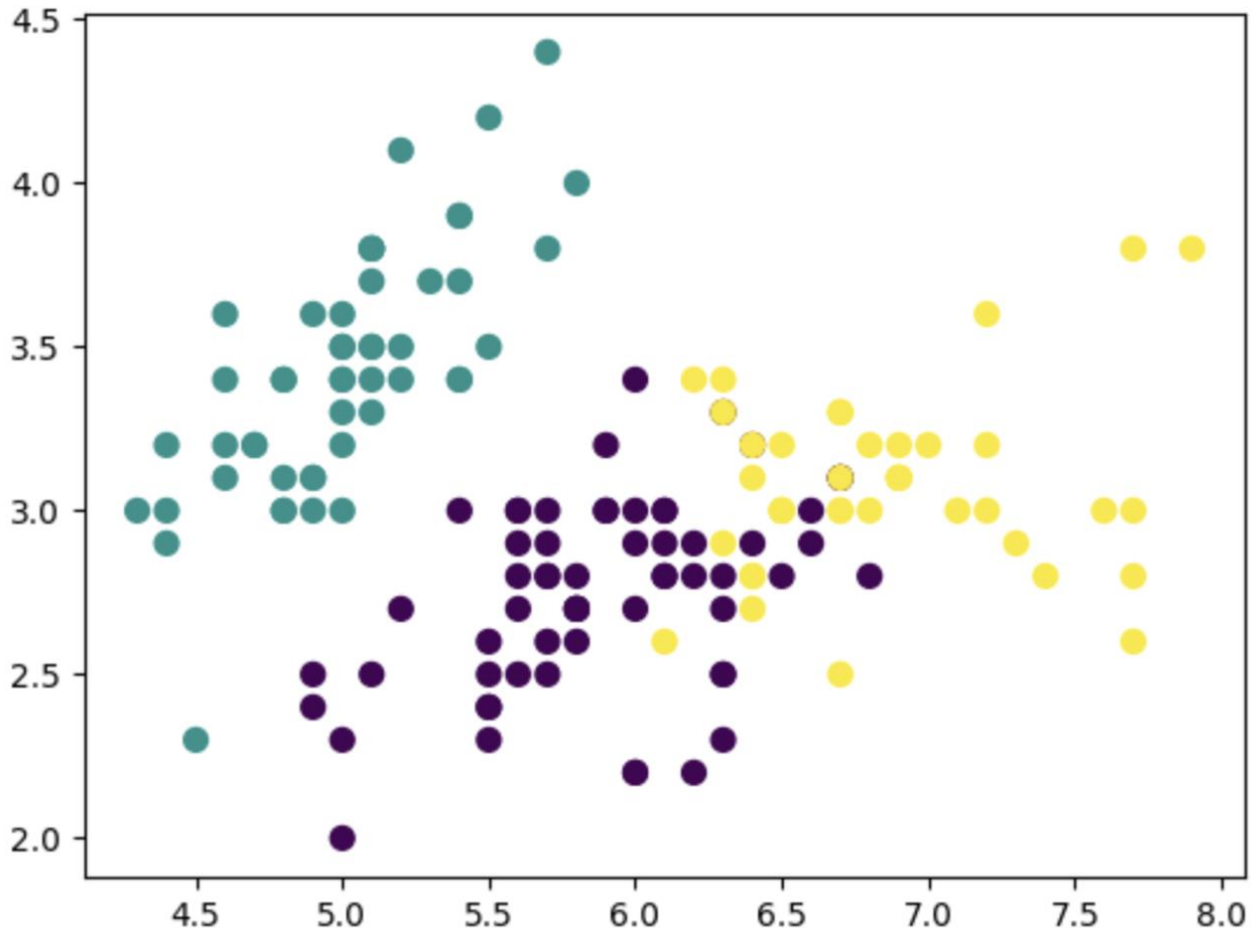
Завдання 2.2. Кластеризація K-середніх для набору даних Iris

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

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

# Візуалізація даних
kmeans = KMeans(
    n_clusters=5,
    init="k-means++",
    n_init=10,
    max_iter=300,
    tol=0.0001,
    random_state=None,
    copy_x=True,
)

# Кластеризація даних
kmeans.fit(X)
# Передбачення кластерів
y_kmeans = kmeans.predict(X)
# Візуалізація кластерів
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)


# Функція для знаходження кластерів
def find_clusters(X, n_clusters, rseed=2):
    # Рандомізація центрів кластерів
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    # Пошук кластерів
    while True:
        # Визначення найближчого центру для кожної точки
        labels = pairwise_distances_argmin(X, centers)
```

```
        # Обчислення нових центрів кластерів
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clusters)])

        # Перевірка на збіжність
        if np.all(centers == new_centers):
            break

        centers = new_centers

    return centers, labels


# Візуалізація кластерів
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(n_clusters=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

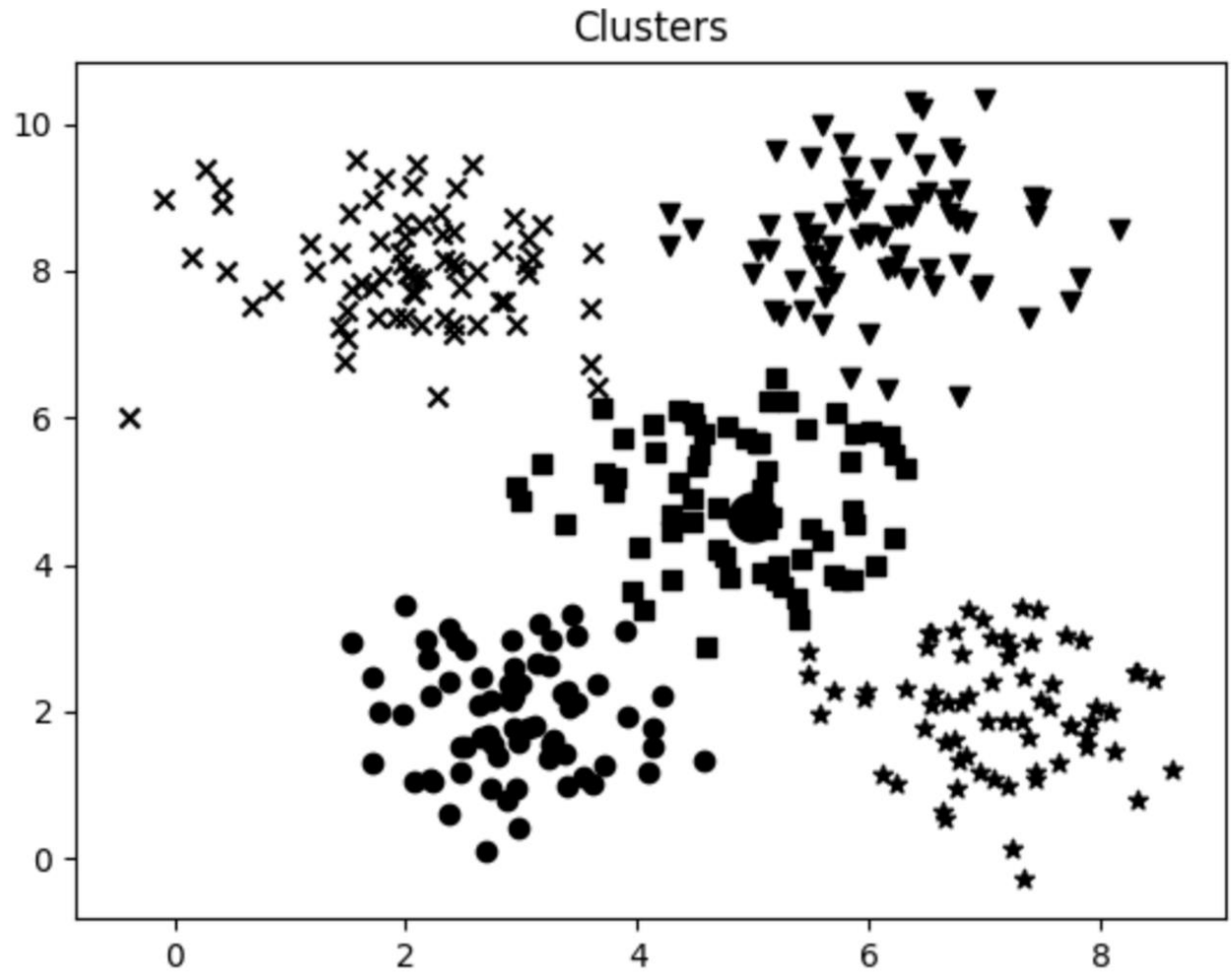


Висновок: Для розподілу даних із набору Iris на групи був використаний метод k-середніх. Результати показали, що алгоритм ефективно класифікує дані за їхньою схожістю. Візуалізація центрів кластерів дозволила детальніше проаналізувати структуру та розподіл точок.

Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього

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

# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')
# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)
```

```
# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_
print("\nCenters of clusters:\n", cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print("\nNumber of clusters in input data =", num_clusters)

# Відображення на графіку точок та центрів кластерів
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",
    )

# Відображення на графіку центру кластера
cluster_center = cluster_centers[i]
plt.plot(
    cluster_center[0],
    cluster_center[1],
    marker="o",
    markerfacecolor="black",
    markeredgecolor="black",
    markersize=15,
)
plt.title("Clusters")
plt.show()
```




```
sh-3.2# python3 LR_7_task_3.py
Matplotlib is building the font cache; this may take a moment.

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
sh-3.2#
```

Висновок: Для визначення оптимальної кількості груп у даних був застосований алгоритм MeanShift. Під час аналізу встановлено, що цей метод ефективно визначає кількість кластерів, спираючись на просторовий розподіл точок. Такий підхід є особливо цінним для роботи з даними, структура яких наперед невідома.

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

```
import datetime
import json
```

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

# Завантаження прив'язок символів компаній до їх повних назв
input_file = "company_symbol_mapping.json"
with open(input_file, "r") as f:
    company_symbols_map = json.load(f)

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

# Завантаження архівних даних котирувань за допомогою yfinance
start_date = "2003-07-03"
end_date = "2007-05-04"

quotes = {}
for symbol in symbols:
    try:
        data = yf.download(symbol, start=start_date, end=end_date)
        if not data.empty:
            quotes[symbol] = data
        else:
            print(f"Дані для {symbol} недоступні.")
    except Exception as e:
        print(f"Помилка завантаження даних для {symbol}: {e}")

# Знаходження спільних дат
common_dates = set.intersection(*[set(data.index) for data in quotes.values()])
common_dates = sorted(list(common_dates))

# Вилучення котирувань за спільними датами
valid_symbols = []
opening_quotes = []
closing_quotes = []

for symbol, data in quotes.items():
    try:
        filtered_data = data.loc[common_dates]
        opening_quotes.append(filtered_data["Open"].values)
        closing_quotes.append(filtered_data["Close"].values)
        valid_symbols.append(symbol)
    except KeyError:
        print(f"Дані для {symbol} не збігаються за датами.")

opening_quotes = np.array(opening_quotes)
closing_quotes = np.array(closing_quotes)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
```

```python
# Фільтрація пропущених значень
quotes_diff = quotes_diff[:, ~np.isnan(quotes_diff).any(axis=0)]
X = quotes_diff.T  # Перетворення на 2-вимірний масив

# Уникнення поділу на 0
std_deviation = X.std(axis=0)
std_deviation[std_deviation == 0] = 1
X /= std_deviation

# Створення моделі графа
edge_model = covariance.GraphicalLassoCV()

# Навчання моделі
with np.errstate(invalid="ignore"):
    edge_model.fit(X)
_, labels = cluster.affinity_propagation(edge_model.covariance_)

# Вивід кластерів
valid_names = [company_symbols_map[symbol] for symbol in valid_symbols]
for i in range(max(labels) + 1):
    cluster_names = ", ".join(np.array(valid_names)[np.array(labels) == i])
    print(f"Cluster {i + 1} ==> {cluster_names}")
```

```
Cluster 1 ==> Exxon, Chevron, ConocoPhillips, Valero Energy
Cluster 2 ==> Toyota, Ford, Honda, Boeing, Mc Donalds, Apple, SAP, Caterpillar
Cluster 3 ==> Kraft Foods
Cluster 4 ==> Coca Cola, Pepsi, Kellogg, Procter Gamble, Colgate-Palmolive, Kimberly-Clark
Cluster 5 ==> Time Warner, Comcast, Marriott, Wells Fargo, JPMorgan Chase, AIG, American express, Bank of America, Goldman Sachs, Xerox, Wal-Mart, Home Depot, Ryder, DuPont de Nemours
Cluster 6 ==> Microsoft, IBM, HP, Amazon, 3M, General Electrics, Cisco, Texas instruments
Cluster 7 ==> Northrop Grumman, Lookheed Martin, General Dynamics
Cluster 8 ==> Walgreen, CVS
Cluster 9 ==> GlaxoSmithKline, Pfizer, Sanofi-Aventis, Novartis
```

Висновок: Результати кластеризації демонструють логічне групування компаній за галузями, такими як нафта і газ, технології, фінанси та фармацевтика, що вказує на схожість у динаміці їхніх акцій. Унікальні випадки, наприклад, окремий кластер Kraft Foods, відображають специфічність бізнес-моделей чи ринкової поведінки. Такий аналіз може бути корисним для інвесторів, щоб виявляти кореляції між компаніями в межах секторів.

Посилання на Github: https://github.com/vladpalamar/Lab7_Ai.git

Висновки: використав спеціалізовані бібліотеки та мову програмування Python дослідив методи неконтрольованої класифікації даних у машинному навчанні.