ЛАБОРАТОРНА РОБОТА № 7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

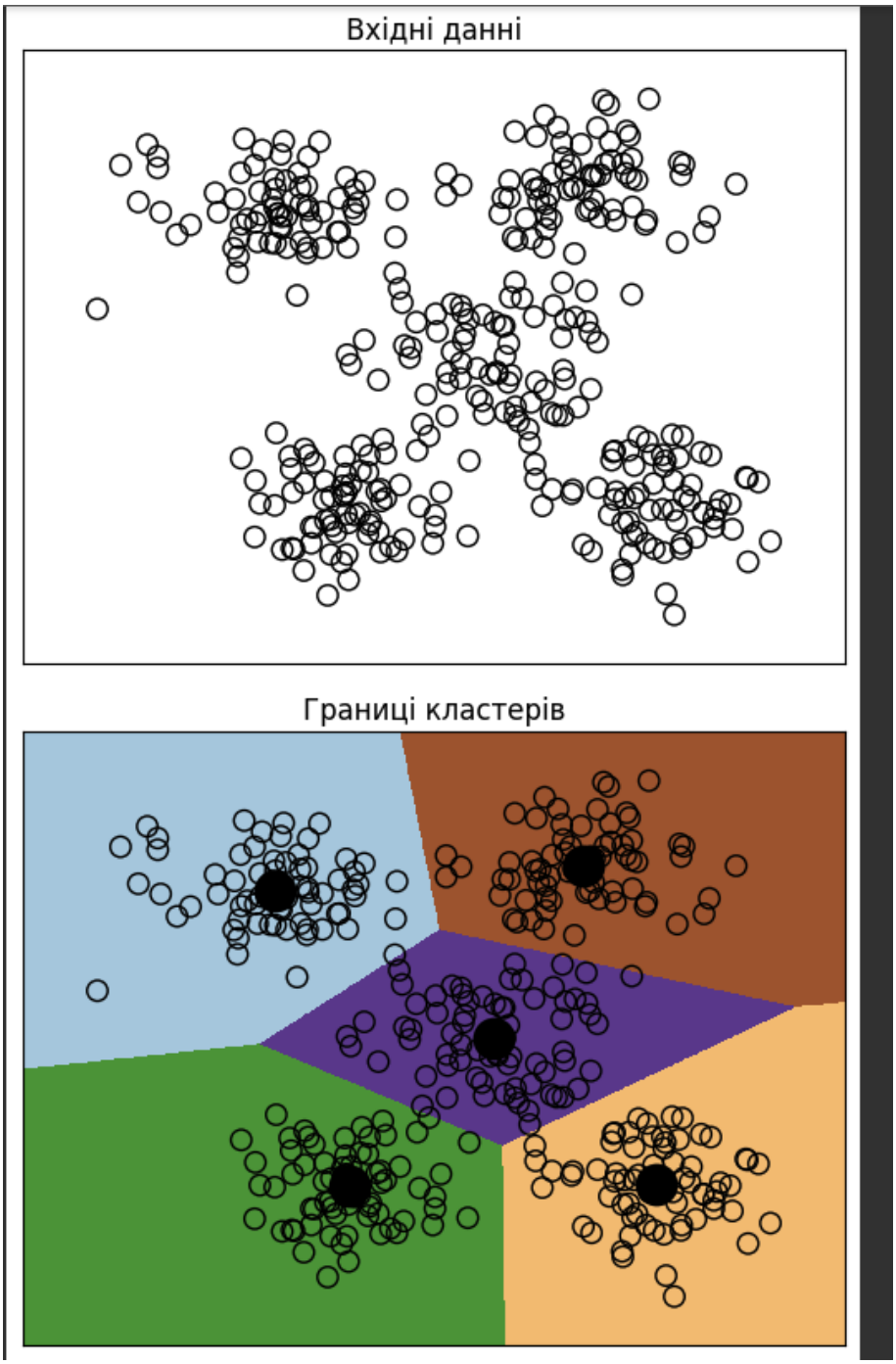
Варіант 1

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
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()
```



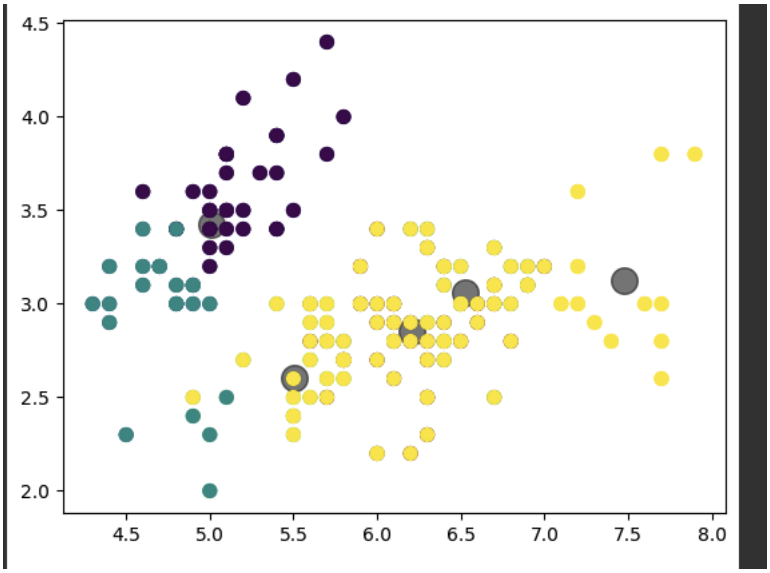
Висновок: Метод k-середніх був ефективно використаний для кластеризації даних, що дозволило розділити вибірку на задану кількість груп. Результати візуалізації наочно демонструють розташування центрів та меж кластерів, підтверджуючи коректність виконаного аналізу.

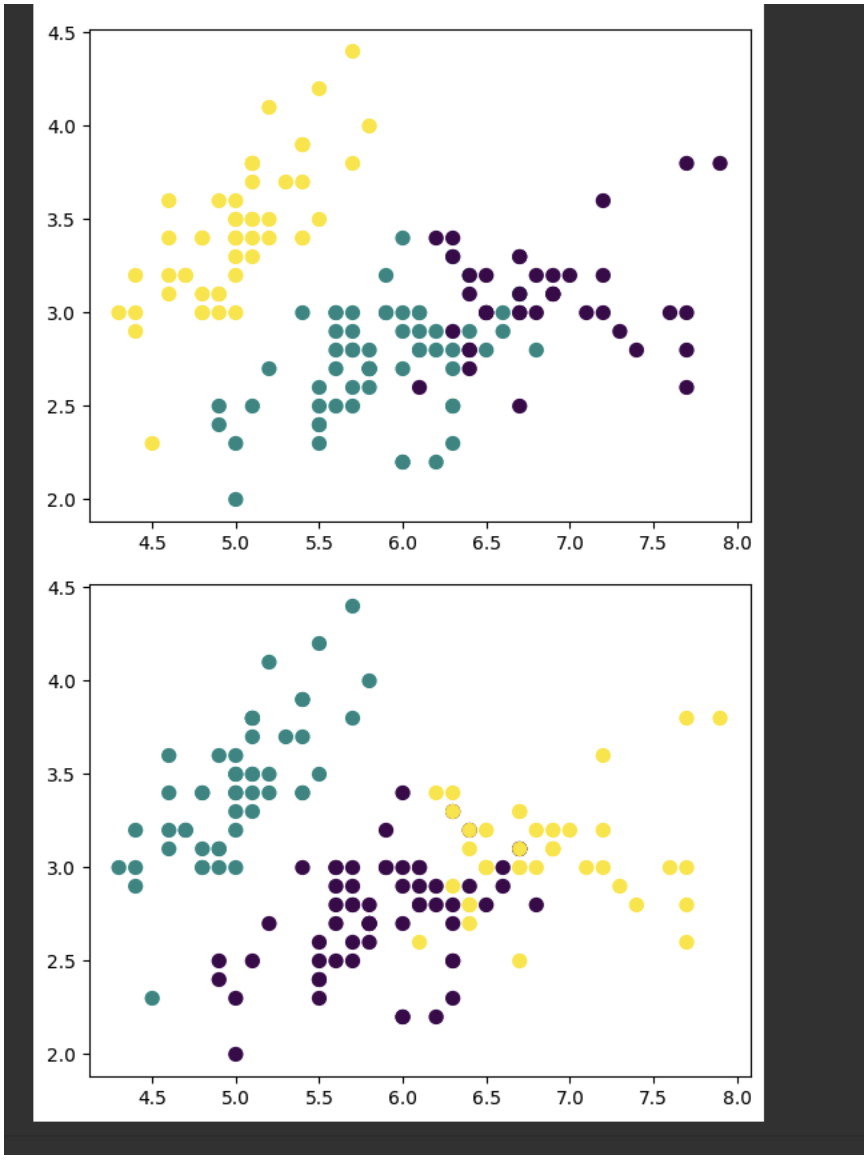
Завдання 2.2. Кластеризація K-середніх для набору даних Iris

```
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()
```





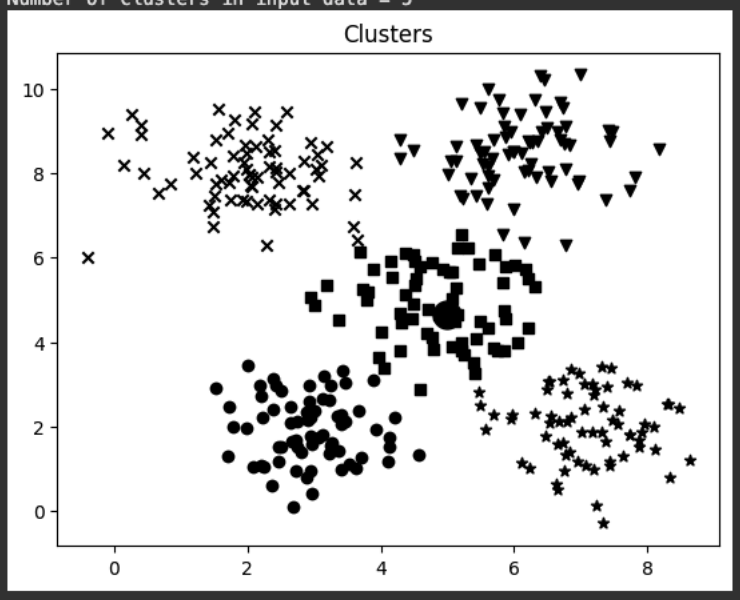
Висновок: Алгоритм k-середніх застосовано для кластеризації даних із набору Iris, що дало змогу поділити їх на групи за подібними ознаками. Отримані результати підтверджують ефективність методу у групуванні даних. Візуалізація центрів кластерів забезпечила детальний аналіз структури та особливостей розподілу 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()
```

```
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
```



Висновок: Метод MeanShift ефективно використано для визначення оптимальної кількості кластерів у даних. Результати підтверджують здатність алгоритму автоматично встановлювати кількість груп, враховуючи просторовий розподіл точок, що робить його дієвим інструментом для аналізу даних із невизначеною структурою.

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

```
import datetime
import json
import numpy as np
from matplotlib.finance import quotes_historical_yahoo_ochl as quotes_yahoo
from sklearn import cluster, covariance

# Вхідний файл із символічними позначеннями компаній
input_file = "company_symbol_mapping.json"

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

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

```
# Завантаження архівних даних котирувань
start_date = datetime.datetime(2003, 7, 3)
end_date = datetime.datetime(2007, 5, 4)
quotes = [quotes_yahoo(symbol, start_date, end_date, asobject=True) for symbol
in symbols]

# Вилучення котирувань, що відповідають
# відкриттю та закриттю біржі
opening_quotes = np.array([quote.open for quote in quotes]).astype(np.float)
closing_quotes = np.array([quote.close for quote in quotes]).astype(np.float)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
X = quotes_diff.copy().T
X /= X.std(axis=0)

# Створення моделі графа
edge_model = covariance.GraphLassoCV()

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

for i in range(max(labels) + 1):
    print("Cluster", i + 1,"==>", ", ".join(names[labels == i]))
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-15-8c7bca9cac92> in <cell line: 4>()
      2 import json
      3 import numpy as np
----> 4 from matplotlib.finance import quotes_historical_yahoo_ochl as quotes_yahoo
      5 from sklearn import cluster, covariance
      6

ModuleNotFoundError: No module named 'matplotlib.finance'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------

OPEN EXAMPLES
```

Код не працюватиме, оскільки модуль matplotlib.finance більше не підтримується.

Посилання на Github:

https://github.com/Vladislaw2533/SHI_Barabash_Vlad_IPZ_21_3

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