Лабораторна робота №7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ

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

Хід роботи:

Завдання 7.1 Кластеризація даних за допомогою методу k-середніх

Лістинг коду:

```
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_cluster = 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 дані")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

# Створення об'єкту KMeans
kmeans = KMeans(init='k-means++', n_clusters=num_cluster, n_init=10)

# Навчання моделі кластеризації KMeans
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()
```

Результат виконання програми:

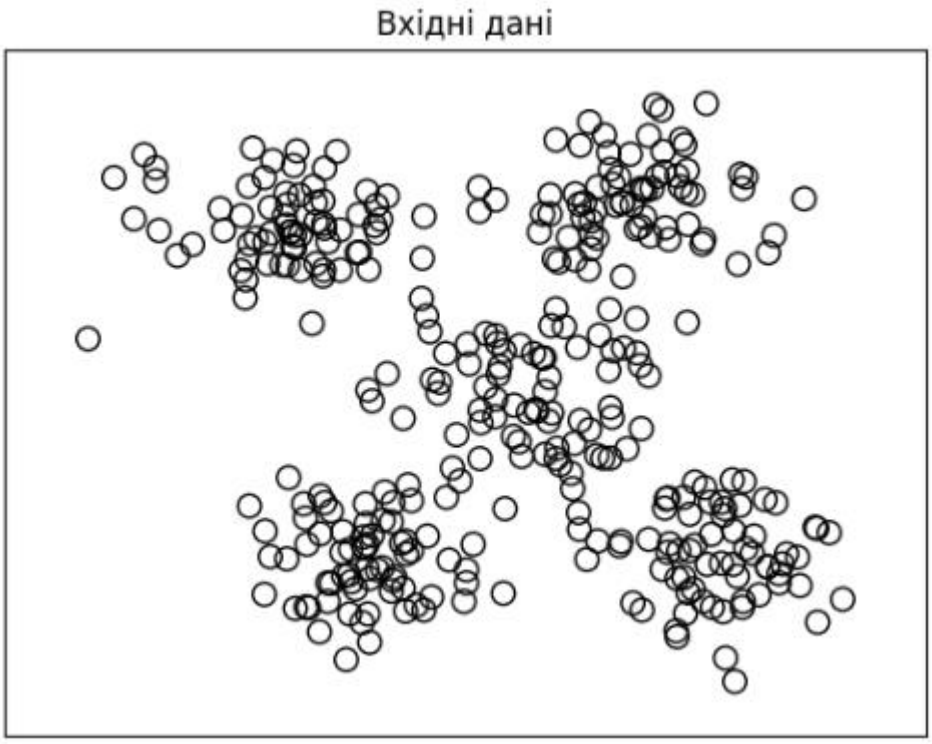

Рисунок 1 - Результат виконання програми

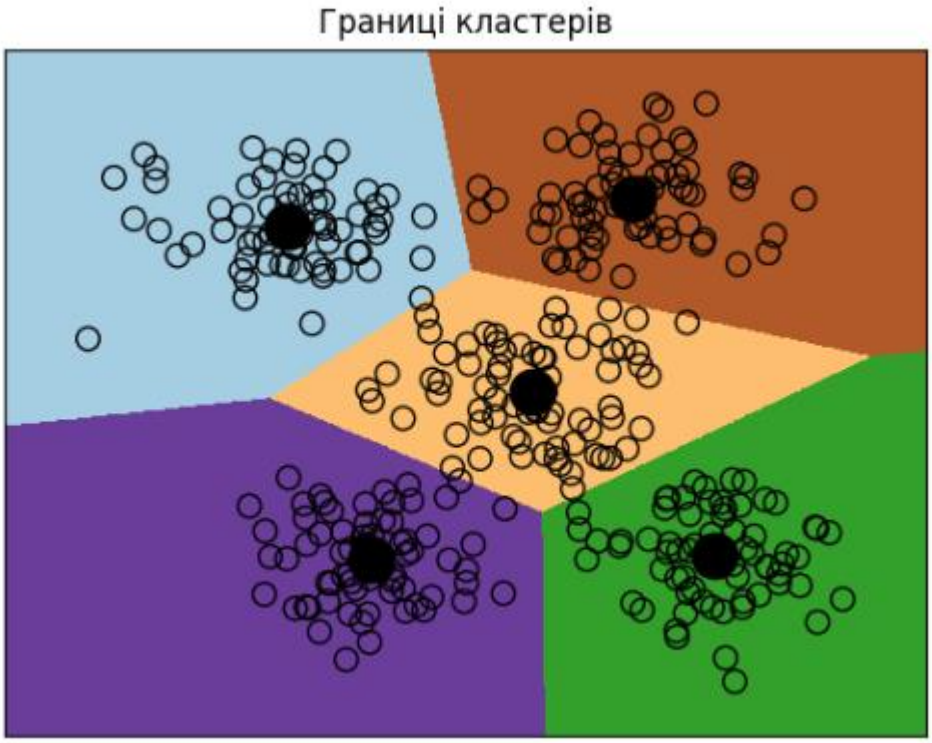

Рисунок 2 - Результат виконання програми

Висновки:

1. **Результати кластеризації**:
   - Дані були успішно розділені на 5 кластерів.
   - Межі кластерів візуалізовані на другому графіку.
2. **Інтерпретація графіків**:
   - На першому графіку видно структуру вхідних даних (групи точок).
   - Другий графік демонструє чіткі межі між кластерами, визначені алгоритмом KMeans. Центри кластерів (чорні точки) розташовані всередині відповідних кластерів.
3. **Коректність роботи алгоритму**:
   - Судячи з графіків, алгоритм KMeans ефективно виконав свою задачу, чітко відокремивши кластери.

Завдання 7.2 Кластеризація K-середніх для набору даних Iris

Лістинг коду:

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

```
# Завантаження даних
iris = load_iris()
X = iris.data  # Ознаки
y = iris.target  # Мітки класів

# Ініціалізація моделі KMeans з 8 кластерами
kmeans = KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300,
tol=0.0001, verbose=0, random_state=None, copy_x = True)

# Навчання моделі на вхідних даних
kmeans.fit(X)

# Прогноз кластерів для кожного зразка
y_kmeans = kmeans.predict(X)

# Візуалізація результатів кластеризації
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)

# Визначення функції для пошуку кластерів
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:
        # Призначення точок до кластерів
        labels = pairwise_distances_argmin(X, centers)
        # Оновлення центрів кластерів
        new_centers = np.array([X[labels == i].mean(0)
        for i in range(n_clusters)])
        # Перевірка умови завершення
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels

# Визначення центрів кластерів та міток для них
centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

# Визначення центрів кластерів та міток для них з іншим random_state
centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

# Використання KMeans з бібліотеки scikit-learn для кластеризації
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

# Відображення графіка
plt.show()
```

Результат виконання програми:

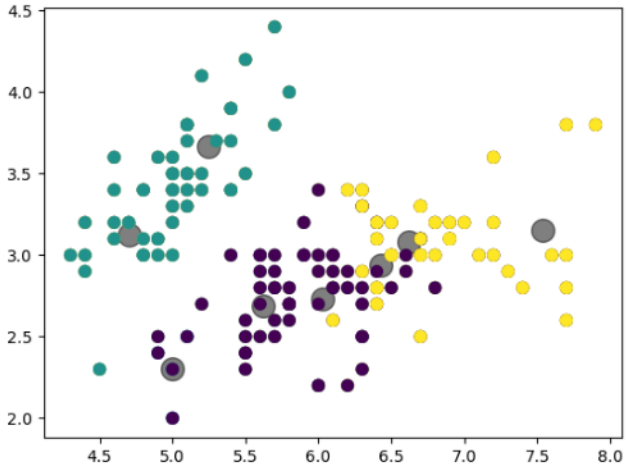

Рисунок 3 - Результат виконання програми

Графік демонструє результати кластеризації набору даних Iris за допомогою алгоритму **K-Means**:

1. 3 кластери є оптимальним вибором, оскільки відповідають трьом реальним класам даних.
2. Кастомна реалізація find_clusters та бібліотечна функція KMeans дають схожі результати, із чітким розділенням між групами.
3. Вибір початкових центрів (через random_state) впливає на проміжні результати, але остаточний розподіл кластерів залишається стабільним.

Завдання 7.3 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

# 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)

# Витягнення 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):
    # Відображення на графіку точок які належать поточному кластеру
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,
color='black')

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

Результат виконання програми:

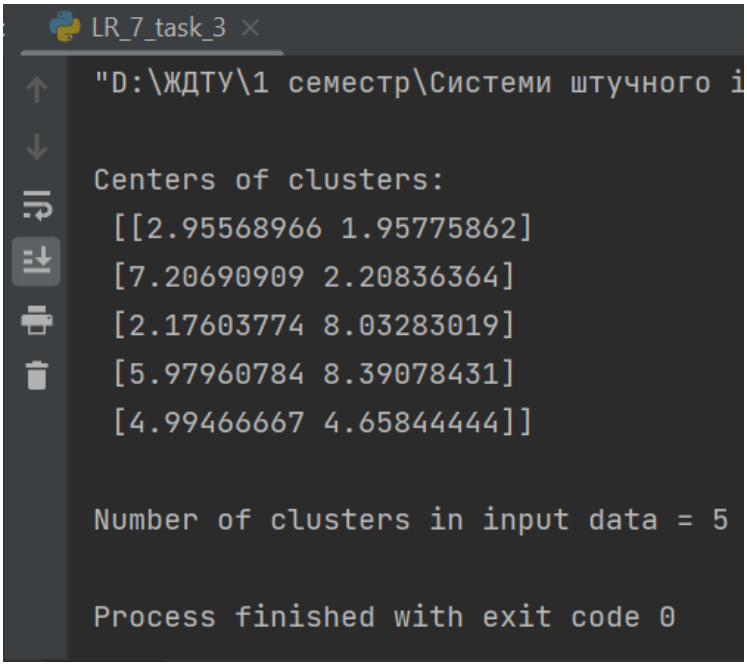

Рисунок 4 - Результат виконання програми

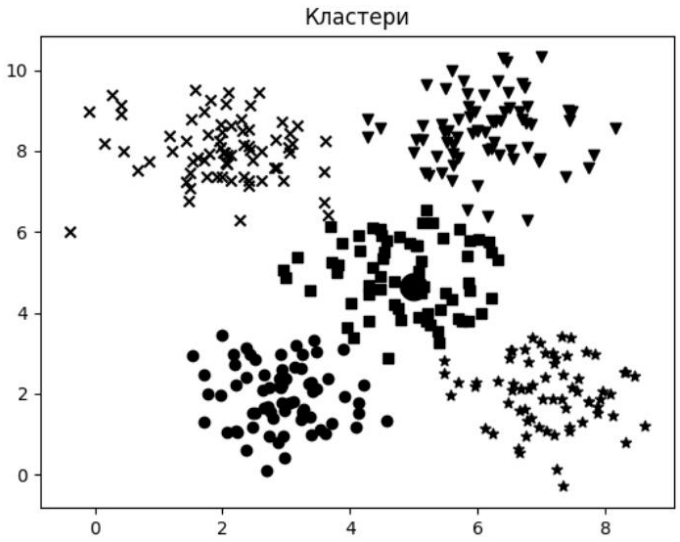

Рисунок 5 - Результат виконання програми

Результати кластеризації методом зсуву середнього дозволяють зробити наступні висновки:

1. **Кількість кластерів**: У даних було виявлено 5 кластерів. Це свідчить про те, що точки у вхідному наборі даних утворюють чітко виражені групи з подібними характеристиками.
2. **Центри кластерів**: Для кожного з кластерів визначено центр, який є середнім значенням координат точок у кластері.
3. **Візуалізація кластерів**: На графіку кожен кластер позначений окремим маркером, що дозволяє легко побачити структуру даних. Центри кластерів виділені більшими маркерами, що робить їх добре помітними.
4. **Алгоритм і параметри**: Алгоритм MeanShift автоматично визначив ширину вікна (bandwidth), що дозволило адаптивно виділити відповідну кількість кластерів.

Алгоритм успішно виконав сегментацію даних. Отримані кластери мають чіткі межі. Візуалізація підтверджує якісну роботу алгоритму, адже точки згруповані природно та без суттєвих перекриттів між кластерами.

Завдання 7.4 Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

Лістинг коду:

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

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

# Завантаження мапи символів компаній
with open(input_file, "r") as f:
    company_symbols_map = json.loads(f.read())

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

# Визначення діапазону дат для історичних котирувань акцій
start_date = datetime.datetime(2003,7,3)
end_date = datetime.datetime(2007,5,4)

# Завантаження історичних котирувань акцій за допомогою yfinance
quotes = []
valid_symbols = []
```

```
for symbol in symbols:
    try:
        data = yf.download(symbol, start=start_date, end=end_date)
        if not data.empty:
            quotes.append(data)
            valid_symbols.append(symbol)
        else:
            print(f"Дані для {symbol} відсутні у вказаний період.")
    except Exception as e:
        print(f"Не вдалося завантажити дані для {symbol}: {e}")

# Перевірка, чи є валідні символи
if not quotes:
    print(
        "Немає валідних даних для жодного символу. Перевірте вашу мапу символів
та наявність даних."
    )
else:
    symbols = valid_symbols  # Оновлюємо символи на валідні

    opening_quotes = []
    closing_quotes = []
    # Видобуття котирувань при відкритті та закритті
    try:
        opening_quotes = np.array([quote["Open"].values for quote in quotes]).T
        closing_quotes = np.array([quote["Close"].values for quote in quotes]).T
    except Exception as e:
        print("Помилка при обробці даних:", e)

    # Обчислення різниці між котируваннями при відкритті та закритті
    quotes_diff = closing_quotes - opening_quotes

    # Нормалізація даних
    X = quotes_diff.copy()
    X /= X.std(axis=0)

    # Створення моделі графу
    edge_model = covariance.GraphicalLassoCV()

    # Навчання моделі
    with np.errstate(invalid="ignore"):
        edge_model.fit(X)

    # Побудова моделі кластеризації з використанням моделі Affinity Propagation
    _, labels = cluster.affinity_propagation(edge_model.covariance_)
    num_labels = labels.max()

    # Виведення результатів кластеризації
    print("\nКластеризація акцій на основі різниці між котируваннями при
відкритті та закритті:\n")
    for i in range(num_labels + 1):
        cluster_indices = np.where(labels == i)[0]
        cluster_names = names[cluster_indices]
        if len(cluster_names) > 0:
            print("Кластер", i + 1, "==>", ", ".join(cluster_names))
```

У завданні до л.р. відсутній файл company_symbol_mapping.json

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

Посилання на GitHub:

https://github.com/AlexanderVolkovIPZ/AIS/tree/master/Lab-7