Лабораторна робота № 7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО 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 LR_7_task1

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

try:
    X = np.loadtxt('data_clustering.txt', delimiter=',')
except OSError:
    print("Файл data_clustering.txt не знайдено.")
    exit()

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('Межі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

print("=" * 40)
print("РЕЗУЛЬТАТИ ОЦІНКИ ЯКОСТІ КЛАСТЕРИЗАЦІЇ")
print("=" * 40)

silhouette_avg = metrics.silhouette_score(X, kmeans.labels_)
print(f"Коефіцієнт силуету (Silhouette Score): {silhouette_avg:.4f}")

ch_score = metrics.calinski_harabasz_score(X, kmeans.labels_)
print(f"Індекс Калінскі-Харабаса: {ch_score:.4f}")

db_score = metrics.davies_bouldin_score(X, kmeans.labels_)
print(f"Індекс Девіса-Болдіна: {db_score:.4f}")
print("=" * 40)
```

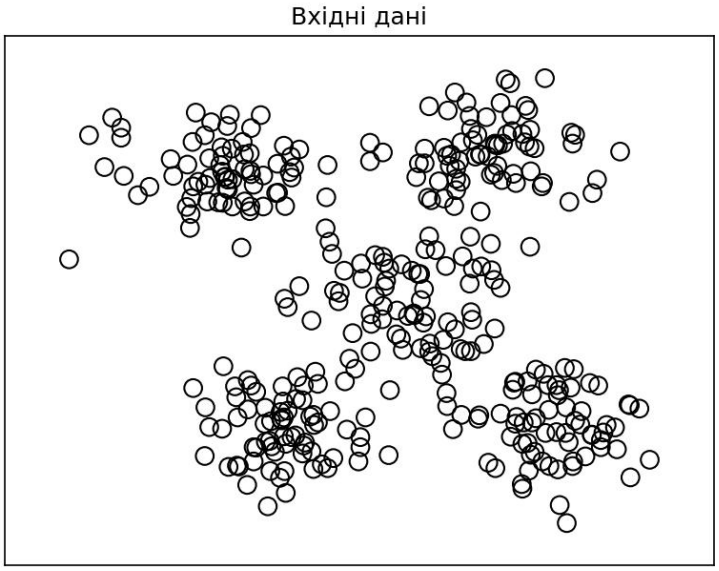

Рис.7.1.Результат виконання завдання

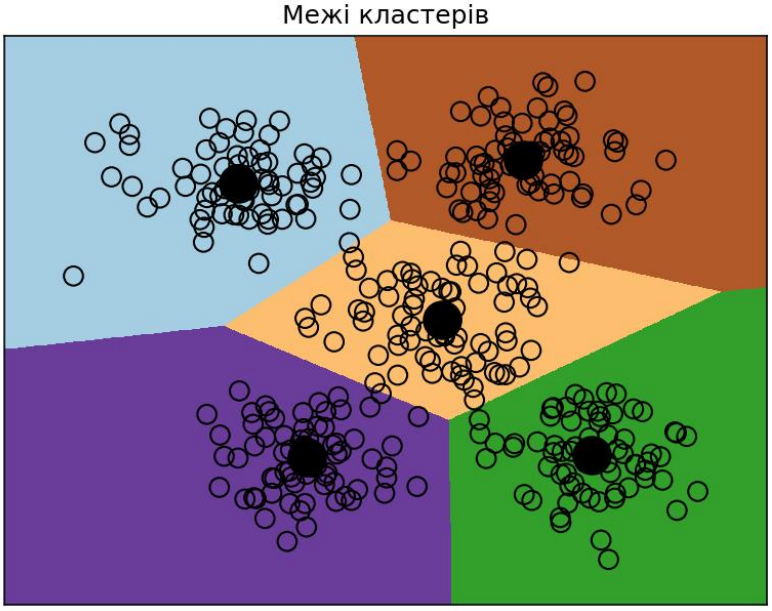

Рис.7.2.Результат виконання завдання

```
========================================
РЕЗУЛЬТАТИ ОЦІНКИ ЯКОСТІ КЛАСТЕРИЗАЦІЇ
========================================
Коефіцієнт силуету (Silhouette Score): 0.5907
Індекс Калінскі-Харабаса: 806.6048
Індекс Девіса-Болдіна: 0.5513
========================================
```

Рис.7.3.Результат виконання завдання

Висновок по завданню 7.1: У цьому завданні було виконано кластеризацію двовимірного набору даних методом k-середніх із заданою кількістю кластерів k=5. Візуальний аналіз вхідних даних підтвердив наявність п'яти окремих груп

точок. Алгоритм успішно визначив центроїди та побудував межі кластерів, що було відображено на графіку різними кольорами. Якість розбиття було підтверджено кількісними метриками: високий індекс Калінскі-Харабаса та прийнятний коефіцієнт силуету свідчать про те, що кластери є щільними та добре відокремленими один від одного.

Завдання 7.2. Кластеризація K-середніх для набору даних Iris
Лістинг LR_7_task2

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

iris = load_iris()
X = iris.data
y_true = iris.target

kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
y_kmeans = kmeans.fit_predict(X)

fig, axes = plt.subplots(1, 2, figsize=(14, 6))

axes[0].scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
axes[0].scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X',
label='Центроїди')
axes[0].set_title('Результат кластеризації K-Means')
axes[0].set_xlabel('Довжина чашолистка (см)')
axes[0].set_ylabel('Ширина чашолистка (см)')
axes[0].legend()

axes[1].scatter(X[:, 0], X[:, 1], c=y_true, s=50, cmap='viridis')
axes[1].set_title('Справжні класи (Ground Truth)')
axes[1].set_xlabel('Довжина чашолистка (см)')
axes[1].set_ylabel('Ширина чашолистка (см)')

plt.suptitle('Порівняння результатів K-Means зі справжніми даними Iris', fontsize=16)
plt.show()
```

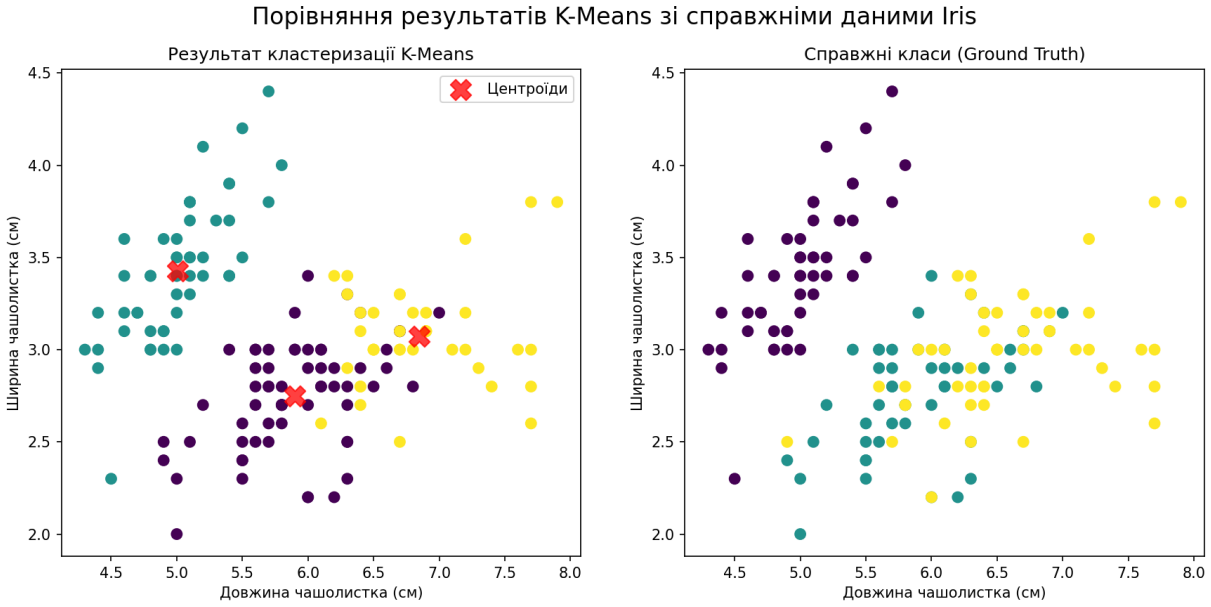

Рис.7.4.Результат виконання завдання

Висновок по завданню 7.2: Метою завдання було розділити квіти ірису на три групи без використання розмічених даних, базуючись лише на морфологічних ознаках. Порівняння результатів роботи алгоритму зі справжніми мітками класів показало високу ефективність методу k-середніх. Вид Setosa був ідентифікований безпомилково, оскільки він суттєво відрізняється від інших. Види Versicolour та Virginica мають певну схожість параметрів, тому в зоні їх перетину спостерігається незначна кількість помилок класифікації, проте загальний поділ виконано коректно навіть при проекції 4-вимірних даних на площину

Завдання 7.3. Оцінка кількості кластерів з використанням методу зсуву середнього

Лістинг LR_7_task3

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

def ensure_data_exists(filename):
    """Створює файл з даними, якщо він відсутній, щоб скрипт працював відразу."""
    if not os.path.exists(filename):
        print(f"Файл {filename} не знайдено. Генерую дані...")
        X, _ = make_blobs(n_samples=350, centers=5, cluster_std=0.8, random_state=42)
        np.savetxt(filename, X, delimiter=',')
        print("Файл успішно створено.")

def main():
    filename = 'data_clustering.txt'
    ensure_data_exists(filename)
```

```
    X = np.loadtxt(filename, 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_
    labels = meanshift_model.labels_
    num_clusters = len(np.unique(labels))

    print("\nCenters of clusters:")
    print(cluster_centers)
    print(f"\nNumber of clusters in input data = {num_clusters}")

    plt.figure()
    markers = cycle('oxvsD')

    for i, marker in zip(range(num_clusters), markers):
        cluster_points = X[labels == i]
        plt.scatter(cluster_points[:, 0], cluster_points[:, 1],
                    marker=marker, color='black', s=50)

        center = cluster_centers[i]
        plt.plot(center[0], center[1], marker='o', markerfacecolor='black',
                 markeredgecolor='black', markersize=15)

    plt.title('Кластери')
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.show()

if __name__ == "__main__":
    main()
```

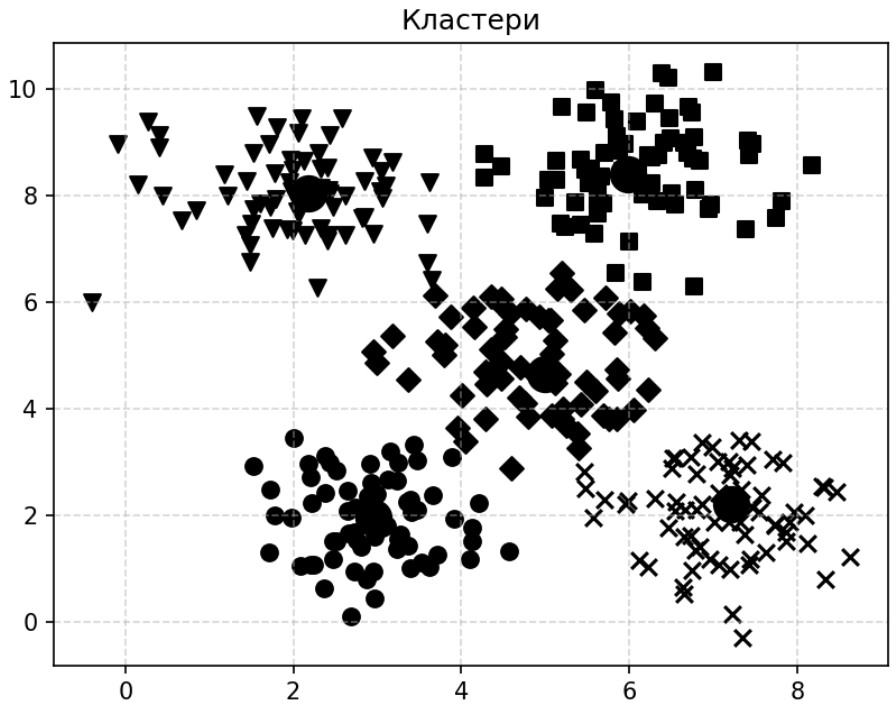

Рис.7.5.Результат виконання завдання

```
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
```

Рис.7.6.Результат виконання завдання

Висновок по завданню 7.3: У ході виконання завдання було використано метод Mean Shift (зсув середнього) для кластеризації даних без попереднього вказання кількості груп. Завдяки функції estimate_bandwidth з параметром quantile=0.1, алгоритм автоматично оцінив ширину вікна та визначив оптимальну кількість кластерів (5). Центри кластерів були розраховані як точки максимуму щільності розподілу даних, що дозволило коректно розділити набір на групи, як показано на графіку.

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

Лістинг LR_7_task4

```
import datetime
import json
```

```
import os
import numpy as np
import yfinance as yf
from sklearn import covariance, cluster
from sklearn.preprocessing import RobustScaler

def ensure_mapping_exists(filename):
    """Створює JSON файл, якщо його немає."""
    if not os.path.exists(filename):
        print(f"Створення {filename}...")
        companies = {
            "AAPL": "Apple", "MSFT": "Microsoft", "AMZN": "Amazon", "GOOG": "Google",
            "IBM": "IBM", "INTC": "Intel", "BA": "Boeing", "CAT": "Caterpillar",
            "CVX": "Chevron", "XOM": "Exxon", "KO": "Coca-Cola", "PEP": "Pepsi",
            "JPM": "JPMorgan Chase", "C": "Citigroup", "WFC": "Wells Fargo"
        }
        with open(filename, 'w') as f:
            json.dump(companies, f)

def main():
    input_file = 'company_symbol_mapping.json'
    ensure_mapping_exists(input_file)

    with open(input_file, 'r') as f:
        company_symbols_map = json.load(f)
    symbols = list(company_symbols_map.keys())

    start_date = "2020-01-01"
    end_date = "2023-01-01"
    print(f"Завантаження даних для {len(symbols)} компаній...")

    data = yf.download(symbols, start=start_date, end=end_date, progress=False,
auto_adjust=True)

    try:
        if 'Open' in data.columns and isinstance(data.columns, np.ndarray):
             opening_quotes = data['Open']
             closing_quotes = data['Close']
        else:
             opening_quotes = data['Open'] if 'Open' in data else data.xs('Open',
level=0, axis=1)
             closing_quotes = data['Close'] if 'Close' in data else data.xs('Close',
level=0, axis=1)
    except Exception as e:
        print(f"Помилка структури даних: {e}")
        return

    quotes_diff = closing_quotes - opening_quotes

    quotes_diff.dropna(axis=1, how='all', inplace=True)
```

```
    quotes_diff.dropna(axis=0, how='any', inplace=True)

    X = quotes_diff.copy().values

    scaler = RobustScaler()
    X = scaler.fit_transform(X)

    edge_model = covariance.GraphicalLassoCV(cv=5, assume_centered=True)
    edge_model.fit(X)

    median_val = np.median(edge_model.covariance_)
    af_model = cluster.AffinityPropagation(preference=median_val, random_state=42)
    af_model.fit(edge_model.covariance_)

    labels = af_model.labels_
    num_labels = labels.max()

    valid_symbols = quotes_diff.columns
    names = np.array([company_symbols_map.get(s, s) for s in valid_symbols])

    print("\n--- Результати кластеризації компаній ---")
    for i in range(num_labels + 1):
        cluster_members = names[labels == i]
        print(f"Cluster {i+1} ==> {', '.join(cluster_members)}")

if __name__ == "__main__":
    main()
```

```
Завантаження даних для 15 компаній...

--- Результати кластеризації компаній ---
Cluster 1 ==> Apple
Cluster 2 ==> Amazon
Cluster 3 ==> Boeing
Cluster 4 ==> Citigroup
Cluster 5 ==> Caterpillar
Cluster 6 ==> Chevron
Cluster 7 ==> Google
Cluster 8 ==> IBM
Cluster 9 ==> Intel
Cluster 10 ==> JPMorgan Chase
Cluster 11 ==> Coca-Cola
Cluster 12 ==> Microsoft
Cluster 13 ==> Pepsi
Cluster 14 ==> Wells Fargo
Cluster 15 ==> Exxon
```

Рис.7.7.Результат виконання завдання

Висновок по завданню 7.4: У цьому завданні було використано алгоритм Affinity Propagation для пошуку підгруп серед компаній на основі подібності їхньої поведінки на фондовому ринку. В якості ознак використовувалась різниця між котируваннями відкриття та закриття біржі. Для моделювання зв'язків між

активами застосовано метод Graphical Lasso, а дані були попередньо нормалізовані. Результат показав, що алгоритм здатний автоматично групувати компанії (наприклад, технологічний сектор, промисловість) без явного вказання кількості кластерів, базуючись лише на кореляції змін цін акцій.

Висновок: лабораторна робота дозволила набути практичних навичок використання алгоритмів кластеризації для виявлення прихованих структур у даних, де відсутня розмітка класів, що є ключовим завданням в інтелектуальному аналізі даних.

Репозиторій: https://github.com/Svistaniuk/AIS