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

2. ЗАВДАННЯ НА ЛАБОРАТОРНУ РОБОТУ ТА МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЙОГО ВИКОНАННЯ

Завдання 2.1. Кластеризація даних за допомогою методу k-середніх

Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data_clustering.txt.

Лістинг:

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

input_file = 'data_clustering.txt'
X = np.loadtxt(input_file, delimiter=',')
num_clusters = 5
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)

kmeans.fit(X)

step_size = 0.01

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))

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')
```

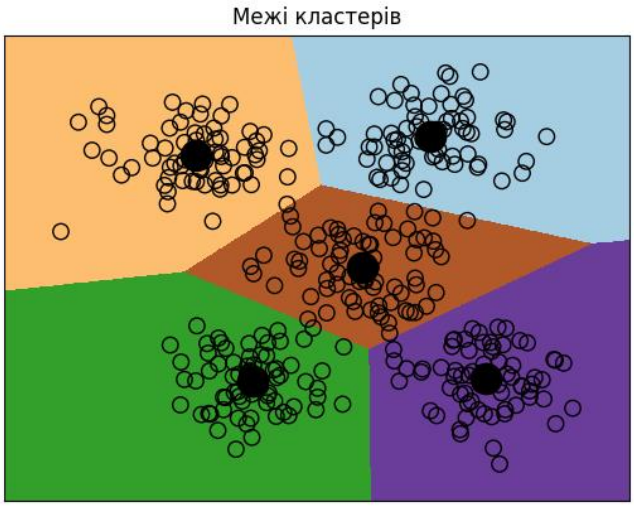
```
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', s=80)

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()

score = metrics.silhouette_score(X, kmeans.labels_, metric='euclidean')
print('Silhouette Score: %.3f' % score)
```

Результат:




```
Silhouette Score: 0.591

Process finished with exit code 0
```

Рис. 1.1-2. Результат кластеризації даних за допомогою методу k-середніх

Реалізовано алгоритм k-means, який успішно розділив вхідні дані на 5 кластерів. Візуалізація меж та отриманий показник якості Silhouette Score (0.591) підтверджують високу точність моделі та чітке розмежування груп.

Завдання 2.2. Кластеризація K-середніх для набору даних Iris

Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина

пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів набору даних Iris.

Лістинг:

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

# Завантаження датасету Iris
iris = load_iris()
X = iris.data
y = iris.target

# Ініціалізація моделі K-Means.
# n_clusters=3 => існує 3 види іриcів.
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300, random_state=0)

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

# Прогнозування кластерів
y_kmeans = kmeans.predict(X)

# Візуалізація результатів (за довжиною та шириною листка)
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis', label='Дані')

# центри кластерів
centers = kmeans.cluster_centers_

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5, label='Центроїди')
plt.title('K-Means clustering on Iris Dataset (sklearn)')
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.legend()
plt.show()

def find_clusters(X, n_clusters, rseed=2):
    # генератор випадкових чисел із заданим зерном для відтворюваності
    rng = np.random.RandomState(rseed)

    # random початкових центрів кластерів із точок даних
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # знаходження найближчого центру для кожної точки
        # обчисленняя відстані та повертає індекс найближчого центру
        labels = pairwise_distances_argmin(X, centers)

        # обчислення середнього значення (mean) для всіх точок кожного кластера
        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

centers, labels = find_clusters(X, 3, rseed=0)
plt.figure(figsize=(10, 6))
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title('K-Means clustering (Manual Implementation)')
```

```
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.show()
```

Результат:

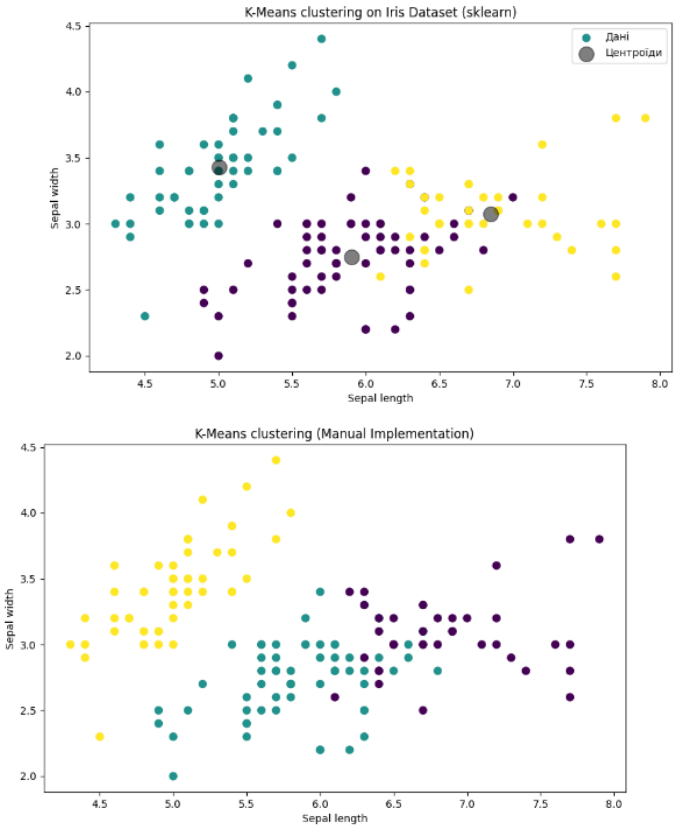


Рис. 2.1. Результат кластеризації K-середніх для набору даних Iris

Обидва підходи (бібліотечний та реалізований) показали ідентичні результати, розділивши набір даних на 3 кластери. Це підтверджує коректність роботи реалізованого вручну алгоритму.

Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього

Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

Лістинг:

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

```
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Оцінка ширини вікна для X
# Чим вищий параметр quantile, тим менше кластерів буде знайдено.
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)

# Витягнемо центри всіх кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)

# Витягнемо кількість 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()
# Маркери для різних кластерів (на 5 кластерів)
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('Кластери')
plt.show()
```

Результат:

```
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

Process finished with exit code 0
```

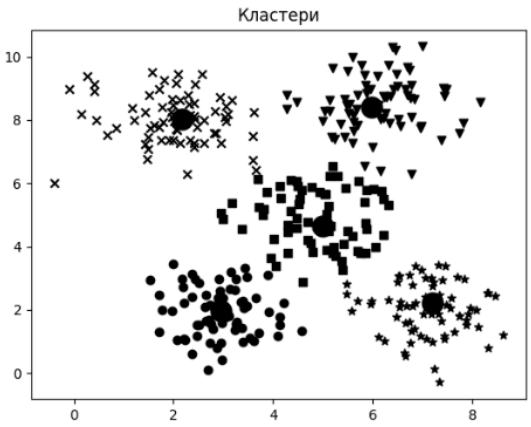


Рис. 3.1. Результат оцінка кількості кластерів методом зсуву середнього

Отримані центри кластерів та їх кількість (5) повністю співпадають з результатами попереднього методу k-means, що підтверджує правильність розбиття даних та надійність алгоритму Mean Shift для аналізу даних з невідомою структурою.

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

Використовуючи модель поширення подібності, знайти підгрупи серед учасників фондового ринку. У якості керуючих ознак будемо використовувати варіацію котирувань між відкриттям і закриттям біржі. Використовувати файл вхідних даних фондового ринку, що доступний в бібліотеці matplotlib. Прив'язки символічних позначень компаній до повних назв містяться у файлі company_symbol_mapping.json.

Лістинг:

```
import datetime
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn import covariance, cluster
import yfinance as yf
import sys
import warnings
import pandas as pd

warnings.filterwarnings("ignore")

input_file = 'company_symbol_mapping.json'
try:
    with open(input_file, 'r') as f:
        company_symbols_map = json.loads(f.read())
except FileNotFoundError:
    print(f"Помилка: Файл {input_file} не знайдено.")
    sys.exit(1)

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

# архівні дані котирувань
start_date = "2003-07-03"
end_date = "2007-05-04"

quotes = []
valid_names = []

print(f"Завантаження даних для {len(symbols)} компаній...")

# Завантажуємо дані пакетом
data_all = yf.download(list(symbols), start=start_date, end=end_date, progress=True,
auto_adjust=True)

# Обробка даних
for symbol, name in zip(symbols, names):
```

```
    try:
        # наявність даних
        if symbol in data_all['Close'] and symbol in data_all['Open']:
            opens = data_all['Open'][symbol].values
            closes = data_all['Close'][symbol].values

            if np.isnan(opens).all() or np.isnan(closes).all():
                print(f"Пропуск {symbol}: відсутні дані.")
                continue

            quotes.append({'Open': opens, 'Close': closes})
            valid_names.append(name)
        else:
            print(f"Пропуск {symbol}: дані не знайдено.")

    except Exception as e:
        print(f"Помилка обробки {symbol}: {e}")

names = np.array(valid_names)

if len(quotes) < 2:
    print("Недостатньо валідних даних для кластеризації.")
    sys.exit(1)

# Обчислення різниці (Variation)
opening_quotes = np.array([q['Open'] for q in quotes]).astype(np.float64)
closing_quotes = np.array([q['Close'] for q in quotes]).astype(np.float64)

# Різниця
quotes_diff = closing_quotes - opening_quotes

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

# модель (Lasso)
print("\nНавчання моделі GraphLassoCV...")
edge_model = covariance.GraphicalLassoCV(cv=5)

with np.errstate(invalid='ignore'):
    edge_model.fit(X)

print("Кластеризація...")
_, labels = cluster.affinity_propagation(edge_model.covariance_, random_state=0)
num_labels = labels.max()

print("\n=== РЕЗУЛЬТАТИ КЛАСТЕРИЗАЦІЇ ===")
for i in range(num_labels + 1):
    cluster_names = names[labels == i]
    print(f"Cluster {i + 1}: {', '.join(cluster_names)}")
```

company_symbol_mapping.json:

```
{
    "XOM": "Exxon",
    "CVX": "Chevron",
    "COP": "ConocoPhillips",
    "MSFT": "Microsoft",
    "IBM": "IBM",
    "AMZN": "Amazon",
    "TM": "Toyota",
    "HMC": "Honda",
    "WHT": "Whitbread",
    "MAR": "Marriott",
    "C": "Citigroup",
    "BAC": "Bank of America",
    "WFC": "Wells Fargo",
    "JPM": "JPMorgan Chase",
    "AIG": "AIG",
    "AXP": "American express",
```

```
    "BA": "Boeing",
    "CAT": "Caterpillar",
    "DD": "DuPont",
    "GE": "General Electric",
    "HD": "Home Depot",
    "INTC": "Intel",
    "KO": "Coca-Cola",
    "MCD": "McDonalds",
    "PEP": "Pepsi",
    "PFE": "Pfizer",
    "PG": "Procter Gamble",
    "UTX": "United Technologies",
    "WMT": "Wal-Mart",
    "XRX": "Xerox",
    "AAPL": "Apple",
    "TXN": "Texas Instruments",
    "GD": "General Dynamics",
    "LMT": "Lockheed Martin"
}
```

Результат:

Рис. 4.1. Результат знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

Cluster 1 (Енергетика): Об'єднав нафтові гіганти Exxon, Chevron та ConocoPhillips.

Cluster 4 (Технології): Згрупував технологічні компанії Microsoft, IBM, Amazon, Intel та Texas Instruments.

Cluster 5 (Напої): Виділив прямих конкурентів Coca-Cola та Pepsi в окрему групу.

Cluster 6 (Оборона та авіація): Об'єднав Boeing, General Dynamics та Lockheed Martin.

Cluster 3 (Фінанси та рітейл): Найбільший кластер, що охопив банківський сектор (JPMorgan, Wells Fargo, Citigroup) та великі конгломерати.

Модель графічного Лассо (GraphicalLassoCV) для побудови матриці подібності та подальша кластеризація методом Affinity Propagation дозволили автоматично відтворити галузеву структуру ринку, базуючись виключно на математичному аналізі коливань цін акцій.

Висновок до роботи: у ході виконання лабораторної роботи було використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено методи неконтрольованої класифікації даних у машинному навчанні.

Репозиторій: https://github.com/MykolaTrofimchuk/AI-systems/tree/main/Lab07