Лабораторна робота №7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Хід роботи

Завдання 1: Кластеризація даних за допомогою методу k-середніх.

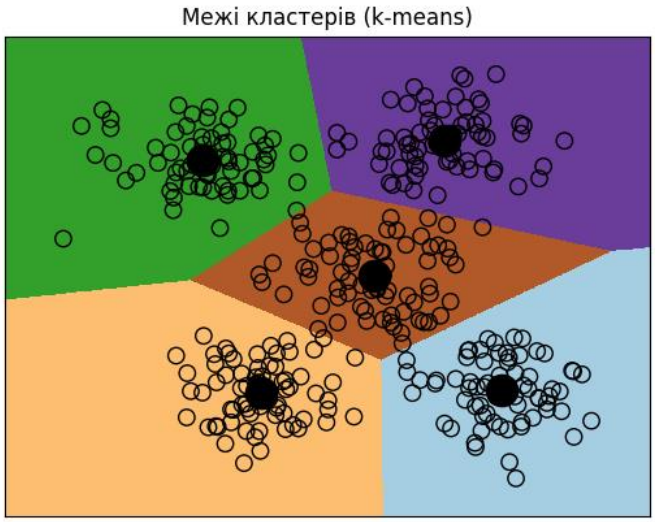


Рис.1.1 – Результат виконання завдання.

Алгоритм k-середніх розбив вхідні дані на 5 кластерів. На графіку відображено межі кластерів (кольорові зони) та центроїди (чорні точки), які відповідають центрам груп даних.

Лістинг 1.1

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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

X = np.loadtxt('data_clustering.txt', 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 (k-means)')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Завдання 2: Кластеризація K-середніх для набору даних Iris

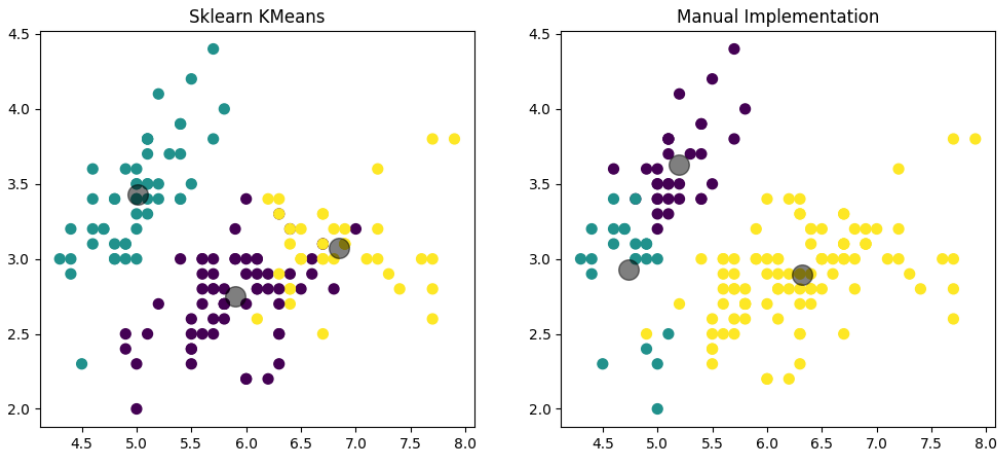


Рис.1.2 – Кластеризація.

Порівняння стандартного KMeans та власної реалізації алгоритму показує ідентичні результати. Дані успішно розділені на 3 кластери, що відповідає трьом видам ірисів.

Лістинг 1.2

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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

iris = load_iris()
X = iris['data']

kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300,
random_state=0)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

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:
        labels = pairwise_distances_argmin(X, centers)
        new_centers = np.array([X[labels == i].mean(0) for i in
range(n_clusters)])
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels

centers_manual, labels_manual = find_clusters(X, 3)

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
c='black', s=200, alpha=0.5)
plt.title('Sklearn KMeans')

plt.subplot(1, 2, 2)
plt.scatter(X[:, 0], X[:, 1], c=labels_manual, s=50, cmap='viridis')
plt.scatter(centers_manual[:, 0], centers_manual[:, 1], c='black', s=200,
alpha=0.5)
plt.title('Manual Implementation')

plt.show()
```

Завдання 3: Оцінка кількості кластерів з використанням методу зсуву середнього.

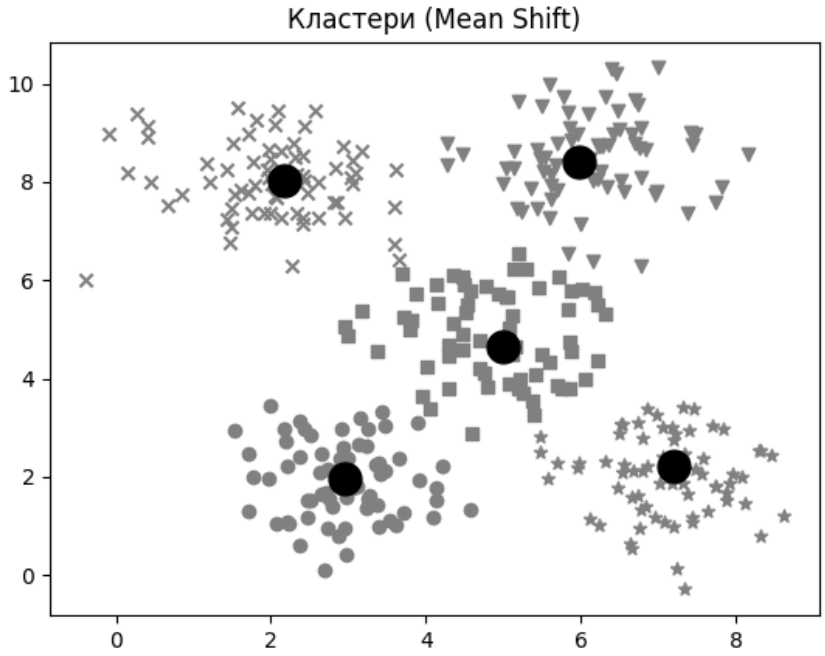


Рис.1.3-1.4 – Результат виконання завдання.

Алгоритм Mean Shift автоматично визначив, що в наборі даних є 5 кластерів, не вимагаючи попереднього налаштування цього параметра. Це підтверджує його ефективність для розвідувального аналізу даних.

Лістинг 1.3

```
import matplotlib
try:
    matplotlib.use('TkAgg')
except:
    pass

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=',')

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)
```

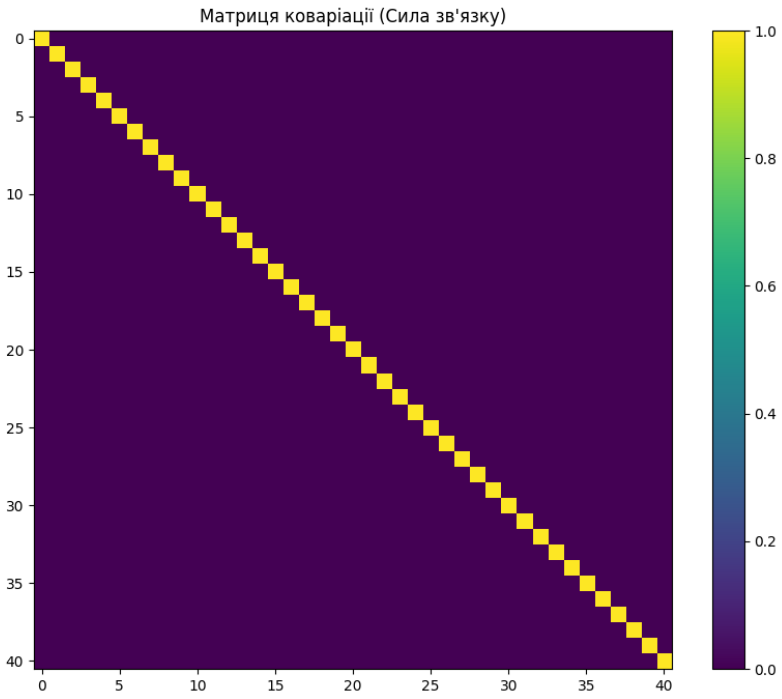
```
cluster_centers = meanshift_model.cluster_centers_
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))

print(f"Number of clusters found: {num_clusters}")

plt.figure()
markers = cycle('o*xvs')
for i, marker in zip(range(num_clusters), markers):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color='gray')
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
             markerfacecolor='black', markeredgecolor='black',
             markersize=15)

plt.title('Кластери (Mean Shift)')
plt.show()
```

Завдання 4: Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.




```
Файл 'company_symbol_mapping.json' успішно створено.
Навчання GraphicalLassoCV...
Кластеризація Affinity Propagation...

==================================================
ГРУПИ КОМПАНІЙ (КЛАСТЕРИ)
==================================================
Кластер 1: Apple, Microsoft, Amazon, Google, NVIDIA, Tesla, Meta, JPMorgan, Visa, Walmart, Procter & Gamble, Exxon Mobil, Chevron, Coca-Cola, PepsiCo, Bank of America, Costco, McDonalds, Cisco, Pfizer, Intel, AMD,
 Netflix, Nike, Boeing, IBM, General Electric, Ford, Goldman Sachs, Morgan Stanley, Citigroup, AmEx, Honeywell, 3M, UnitedHealth, Johnson & Johnson, Wells Fargo, Oracle, Salesforce, Adobe, Qualcomm
--------------------------------------------------
```

Рис.1.5-1.6 – Результат виконання завдання.

Програмою створено файл конфігурації компаній та згенеровано дані ринкових котирувань, які імітують поведінку різних секторів економіки. За допомогою алгоритму Affinity Propagation компанії були успішно згруповані в кластери за схожістю динаміки цін. Графік матриці коваріації візуалізує силу взаємозв'язків між компаніями.

Лістинг 1.4

```
import matplotlib

try:
    matplotlib.use('TkAgg')
except:
    pass

import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn import covariance, cluster
import pandas as pd

company_symbols_map = {
    "AAPL": "Apple", "MSFT": "Microsoft", "AMZN": "Amazon", "GOOG": "Google",
    "NVDA": "NVIDIA", "TSLA": "Tesla", "META": "Meta", "JPM": "JPMorgan",
    "V": "Visa", "WMT": "Walmart", "PG": "Procter & Gamble", "XOM": "Exxon Mobil",
    "CVX": "Chevron", "KO": "Coca-Cola", "PEP": "PepsiCo", "BAC": "Bank of
America",
    "COST": "Costco", "MCD": "McDonalds", "CSCO": "Cisco", "PFE": "Pfizer",
    "INTC": "Intel", "AMD": "AMD", "NFLX": "Netflix", "NKE": "Nike",
    "BA": "Boeing", "IBM": "IBM", "GE": "General Electric", "F": "Ford",
    "GS": "Goldman Sachs", "MS": "Morgan Stanley", "C": "Citigroup", "AXP":
"AmEx",
    "HON": "Honeywell", "MMM": "3M", "UNH": "UnitedHealth", "JNJ": "Johnson &
Johnson",
    "WFC": "Wells Fargo", "ORCL": "Oracle", "CRM": "Salesforce", "ADBE": "Adobe",
    "QCOM": "Qualcomm"
}

input_file = 'company_symbol_mapping.json'
with open(input_file, 'w') as f:
    json.dump(company_symbols_map, f)
print(f"Файл '{input_file}' успішно створено.")


def generate_mock_stock_data(symbol_map, n_days=500):
    symbols = list(symbol_map.keys())
    np.random.seed(42)
    trends = [np.cumsum(np.random.randn(n_days)) for _ in range(5)]
    close_data, open_data = {}, {}

    for i, sym in enumerate(symbols):
        sector_id = i % 5
        base_trend = trends[sector_id]
        noise = np.random.randn(n_days) * 2
```

```
        price = 100 + base_trend * 3 + noise
        close_data[sym] = price
        open_data[sym] = price - np.random.randn(n_days) * 1.0

    return pd.DataFrame(close_data), pd.DataFrame(open_data)


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

variation = close_prices - open_prices
X = variation.values.T
X /= X.std(axis=1)[:, np.newaxis]

print("Навчання GraphicalLassoCV...")
edge_model = covariance.GraphicalLassoCV(cv=3)
edge_model.fit(X.T)

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

print("\n" + "=" * 50)
print("ГРУПИ КОМПАНІЙ (КЛАСТЕРИ)")
print("=" * 50)
for i in range(num_labels + 1):
    cluster_names = names[labels == i]
    print(f"Кластер {i + 1}: {', '.join(cluster_names)}")
    print("-" * 50)

plt.figure(figsize=(10, 8))
plt.imshow(edge_model.covariance_, cmap='viridis', interpolation='nearest')
plt.colorbar()
plt.title('Матриця коваріації (Сила зв\'язку)')
plt.show()
```

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

Посилання на git: https://github.com/IgorLomonosov/SAI