ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

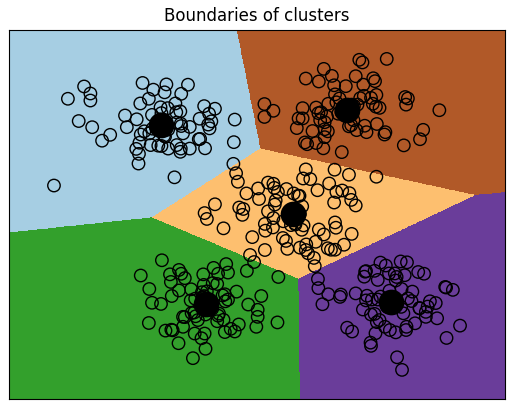
Хід роботи

Завдання 2.1. Кластеризація даних за допомогою методу k-середніх.

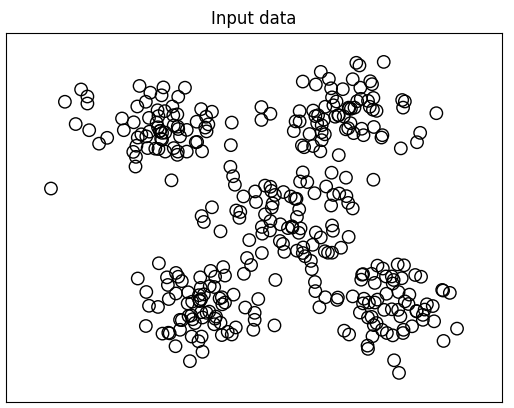
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
num\_clusters = 5  
  
  
plt.figure()  
plt.scatter(X[:,0], X[:,1], marker='o', facecolors='none',edgecolors='black', 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('Input data')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
  
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')  
  
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('Boundaries of clusters')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

**Рис. 1.1 Код програми**



**Рис. 1.2 Результат виконання програми**



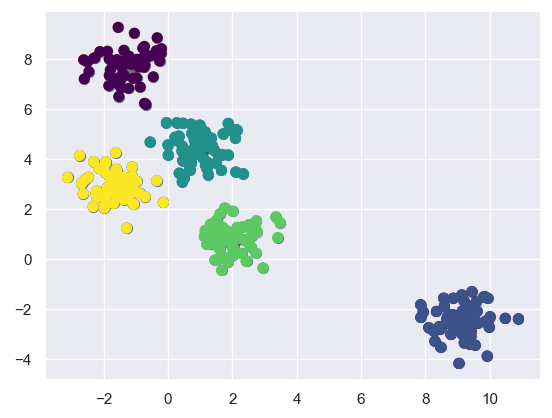
**Рис. 1.3 Результат виконання програми**

Завдання 2.2. Кластеризація K-середніх для набору даних Iris

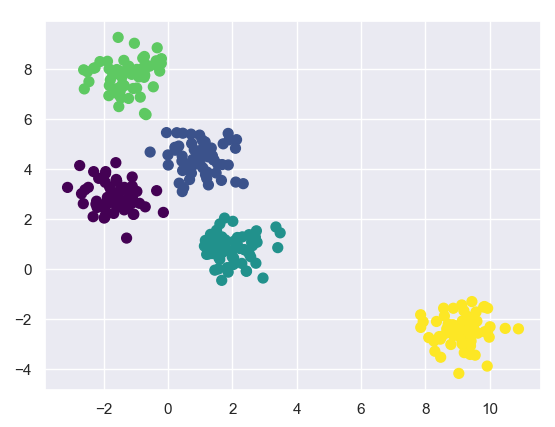
import matplotlib.pyplot as plt  
import seaborn as sns; sns.set()  
import numpy as np  
from sklearn.datasets import make\_blobs  
from sklearn.cluster import KMeans  
  
X, y\_true = make\_blobs(n\_samples=300, centers=5, cluster\_std=0.60, random\_state=0)  
  
plt.scatter(X[:, 0], X[:, 1], s=50)  
  
# Створення об'єкту КМеаns  
kmeans = KMeans(n\_clusters=5)  
  
# Навчання моделі кластеризації КМеаns  
kmeans.fit(X)  
y\_kmeans = kmeans.predict(X)  
  
# Відображення вхідних точок  
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)  
  
from sklearn.metrics import pairwise\_distances\_argmin

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  
  
  
# Відображення центрів кластерів  
centers, labels = find\_clusters(X, 5)  
plt.scatter(X[:, 0], X[:, 1], c=labels,  
 s=50, cmap='viridis')  
plt.show()  
centers, labels = find\_clusters(X, 5, rseed=0)  
plt.scatter(X[:, 0], X[:, 1], c=labels,  
 s=50, cmap='viridis')  
plt.show()  
labels = KMeans(5, random\_state=0).fit\_predict(X)  
plt.scatter(X[:, 0], X[:, 1], c=labels,  
 s=50, cmap='viridis')  
plt.show()

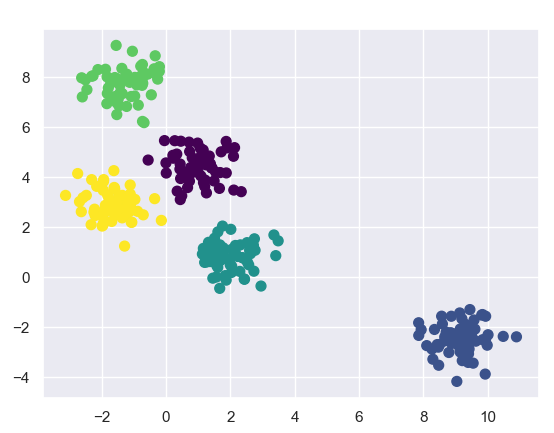
**Рис. 1.4 Код програми**



**Рис. 1.5 Результат виконання програми**



**Рис. 1.6 Результат виконання програми**



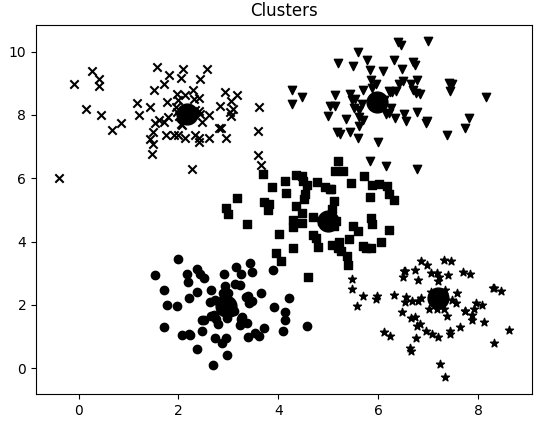
**Рис. 1.7 Результат виконання програми**

Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву

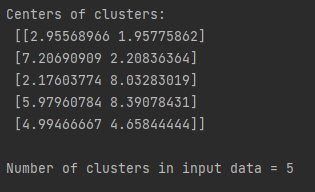
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
bandwidth\_X = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))  
  
meanshift\_model = MeanShift(bandwidth=bandwidth\_X, bin\_seeding=True)  
meanshift\_model.fit(X)  
  
cluster\_centers = meanshift\_model.cluster\_centers\_  
print('\nCenters of clusters:\n', cluster\_centers)  
  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print("\nNumber of clusters in input data =", num\_clusters)  
  
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')

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()

**Рис. 1.8 Код програми**



**Рис. 1.9 Результат виконання програми**



**Рис. 2.1 Результат виконання програми**

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

import datetime  
import json  
  
import numpy as np  
from sklearn import covariance, cluster  
  
from matplotlib.finance import quotes\_historical\_yahoo\_ochl as quotes\_yahoo  
  
input\_file = 'company\_symbol\_mapping.json'  
  
  
with open(input\_file, 'r') as f:  
 company\_symbols\_map = json.loads(f.read())  
  
symbols, names = np.array(list(company\_symbols\_map.items())).T  
  
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]  
  
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)  
  
quotes\_diff = closing\_quotes - opening\_quotes  
  
X = quotes\_diff.copy().T  
X /= X.std(axis=0)  
  
edge\_model = covariance.GraphLassoCV()  
  
with np.errstate(invalid='ignore'):  
 edge\_model.fit(X)  
  
\_, labels = cluster.affinity\_propagation(edge\_model.covariance\_)  
num\_labels = labels.max()  
  
for i in range(num\_labels + 1):  
 print("Cluster", i+1, "==>", ', '.join(names[labels == i]))

**Рис. 2.2 Код програми**

{  
 "TOT": "Total",  
 "XOM": "Exxon",  
 "CVX": "Chevron",  
 "COP": "ConocoPhillips",  
 "VLO": "Valero Energy",  
 "MSFT": "Microsoft",  
 "IBM": "IBM",  
 "TWX": "Time Warner",  
 "CMCSA": "Comcast",  
 "CVC": "Cablevision",  
 "YHOO": "Yahoo",

"DELL": "Dell",  
 "HPQ": "HP",  
 "AMZN": "Amazon",  
 "TM": "Toyota",  
 "CAJ": "Canon",  
 "MTU": "Mitsubishi",  
 "SNE": "Sony",  
 "F": "Ford",  
 "HMC": "Honda",  
 "NAV": "Navistar",  
 "NOC": "Northrop Grumman",  
 "BA": "Boeing",  
 "KO": "Coca Cola",  
 "MMM": "3M",  
 "MCD": "Mc Donalds",  
 "PEP": "Pepsi",  
 "MDLZ": "Kraft Foods",  
 "K": "Kellogg",  
 "UN": "Unilever",  
 "MAR": "Marriott",  
 "PG": "Procter Gamble",  
 "CL": "Colgate-Palmolive",  
 "GE": "General Electrics",  
 "WFC": "Wells Fargo",  
 "JPM": "JPMorgan Chase",  
 "AIG": "AIG",  
 "AXP": "American express",  
 "BAC": "Bank of America",  
 "GS": "Goldman Sachs",  
 "AAPL": "Apple",  
 "SAP": "SAP",  
 "CSCO": "Cisco",  
 "TXN": "Texas instruments",  
 "XRX": "Xerox",  
 "LMT": "Lookheed Martin",  
 "WMT": "Wal-Mart",  
 "WBA": "Walgreen",  
 "HD": "Home Depot",  
 "GSK": "GlaxoSmithKline",  
 "PFE": "Pfizer",  
 "SNY": "Sanofi-Aventis",  
 "NVS": "Novartis",  
 "KMB": "Kimberly-Clark",  
 "R": "Ryder",  
 "GD": "General Dynamics",  
 "RTN": "Raytheon",  
 "CVS": "CVS",  
 "CAT": "Caterpillar",  
 "DD": "DuPont de Nemours"  
}

**Рис. 2.3 Результат виконання програми**

**Висновок:** я використовав спеціалізовані бібліотеки та мову програмування Python дослідивши методи регресії даних у машинному навчанні.

Метод k-середніх (k-means) - це добре відомий алгоритм кластеризації. Його використання передбачає, що кількість кластерів заздалегідь відома. Далі ми сегментуємо дані до підгруп, застосовуючи різні атрибути даних. Ми починаємо з того, що фіксуємо кількість кластерів та, виходячи з цього, класифікуємо дані.

Основна ідея полягає в оновленні положень центроїдів (центрів тяжіння кластеру, або головні точки) на кожній ітерації. Ітеративний процес продовжується до тих пір, поки всі центроїди не займуть оптимального положення.Як неважко здогадатися, у цьому алгоритмі вибір початкового розташування центроїдів відіграє дуже важливу роль, оскільки це безпосередньо впливає на кінцеві результати.

Одна із стратегій полягає в тому, щоб центроїди розташовувалися на якомога більшій відстані один від одного. Базовому методу k-середніх відповідає випадкове розташування центроїдів, тоді як у вдосконаленому варіанті методу (k-means++) ці точки вибираються алгоритмічно з списку вхідних точок даних.