Лабораторна робота №4
Тема: Дослідження методів неконтрольованого навчання

Посилання на гіт хаб:

Завдання 1: Кластерізація данних за допомогою методу к-середніх

Лістинг програми:

```
#Task1
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

#Chart

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(())
#chart add to the report

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)
```

					Житомирська політехніка		
					22.121.06.000 – Лр4		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Медвєдєв В.В.*			Звіт з лабораторної роботи	*Літ.*	*Арк.* 1 / *Аркушів*
Перевір.		Філіпов В.О.					
Керівник						*ФІКТ Гр. ПІ-61*	
Н. контр.							
Зав. каф.							

```
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("Claster Edges")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

plt.xticks(())
plt.yticks(())
plt.show()
#Chart Add to report
```

Після виконання програми отримали наступні два графіки (рис.1) Ям можна бачити за допомогою обраного методу визначення подібності вдалося створити пять кластерів зосередження даних. Визначивши центральні точки підгруп всередені набору даних. Також можна помітити, що деяка кількість точок відноситься до пограничних ділянок й наближені одразу до декількох кластерів.

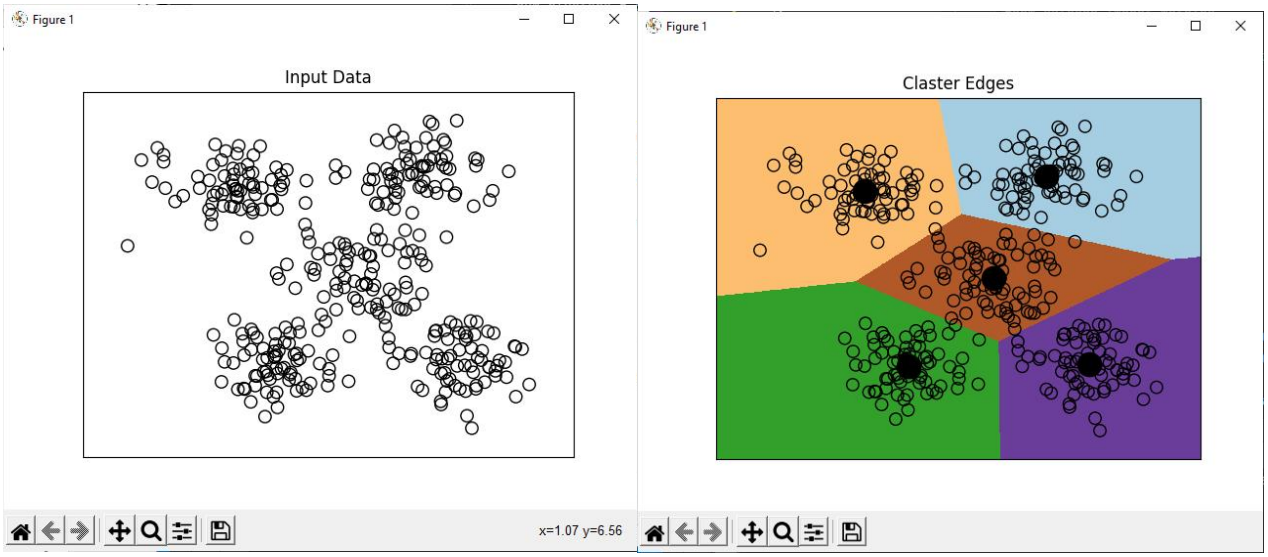

Рисунок 1. Отримані графіки К-середніх

Завдання 2: Кластерізація К-середніх для нобору даних Iris

Використаємо для кластерізації дані з наборів (набір даних іриcів), які вже використовували в попередніх роботах

Лістинг програми:

```
#TASK 2
import numpy as np
import sklearn
from sklearn.svm import SVC
from sklearn.metrics import pairwise_distances_argmin
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics
```

```
from sklearn.datasets import load_iris

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

#отримання налаштованого обїекту к середних
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,
                        algorithm='auto'
                        )

#вчимо прочитаними данними
kmeans.fit(X)

#Передбачте найближчий кластер, до якого належить кожна вибірка в X
y_kmeans = kmeans.predict(X)
#Формуємо діаграми розсіювання y та x із різним розміром та кольором маркера.
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)

#оголошуемо функцію 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:
        #Ця функція обчислює для кожного рядка в X індекс рядка Y, який є найближ-
чим (відповідно до вказаної відстані).
        labels = pairwise_distances_argmin(X, centers)
        #обраховуємо середне значення для цетрів
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clus-
ters)])
        print(centers)
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels


centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

#знаходимо точки центрів для значення минимальної відстані та встановлюємо точки
для діагрмми
centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c = labels, s=50, cmap='viridis')

#Обчислюємо центри кластерів та прогнозуємо індекси кластерів для кожної вибірки
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

plt.show()
```

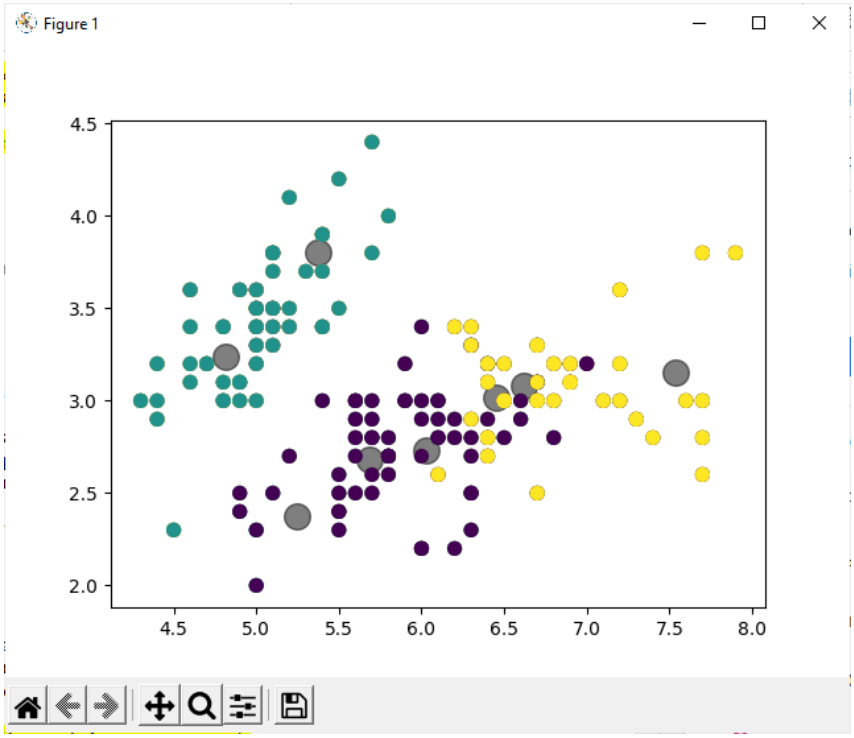

Рисунок 2. Отримана діаграма розсіювання

З використанням тестових даних ірисів вдалося встановити наступну кластеризацію з декількома центрами зосередження кластерів (рис.2)

Завдання 3: Оцінка кількості кластерів з використанням методу зсуву середнього

Лістинг програми:

```
#Task 3 -------
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle

# Завантаження
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)

# Кластеризація даних методом зсуву середнього
cluster_centers = meanshift_model.cluster_centers_
print('\n Centers 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 поточного кластеру
#
    plt.scatter(X[labels==i, 0], X[labels==i, 1], marker=marker, color="black")
# Відображення на графіку центру кластера
    cluster_centers = cluster_centers[i]
    print(cluster_centers[0])
    plt.plot(
        cluster_centers[0],
        cluster_centers[1],
        marker='o',
        markerfacecolor='black',
        markeredgecolor='black',
        markersize=15
    )

    plt.title("Clasterss")
    plt.show()
```

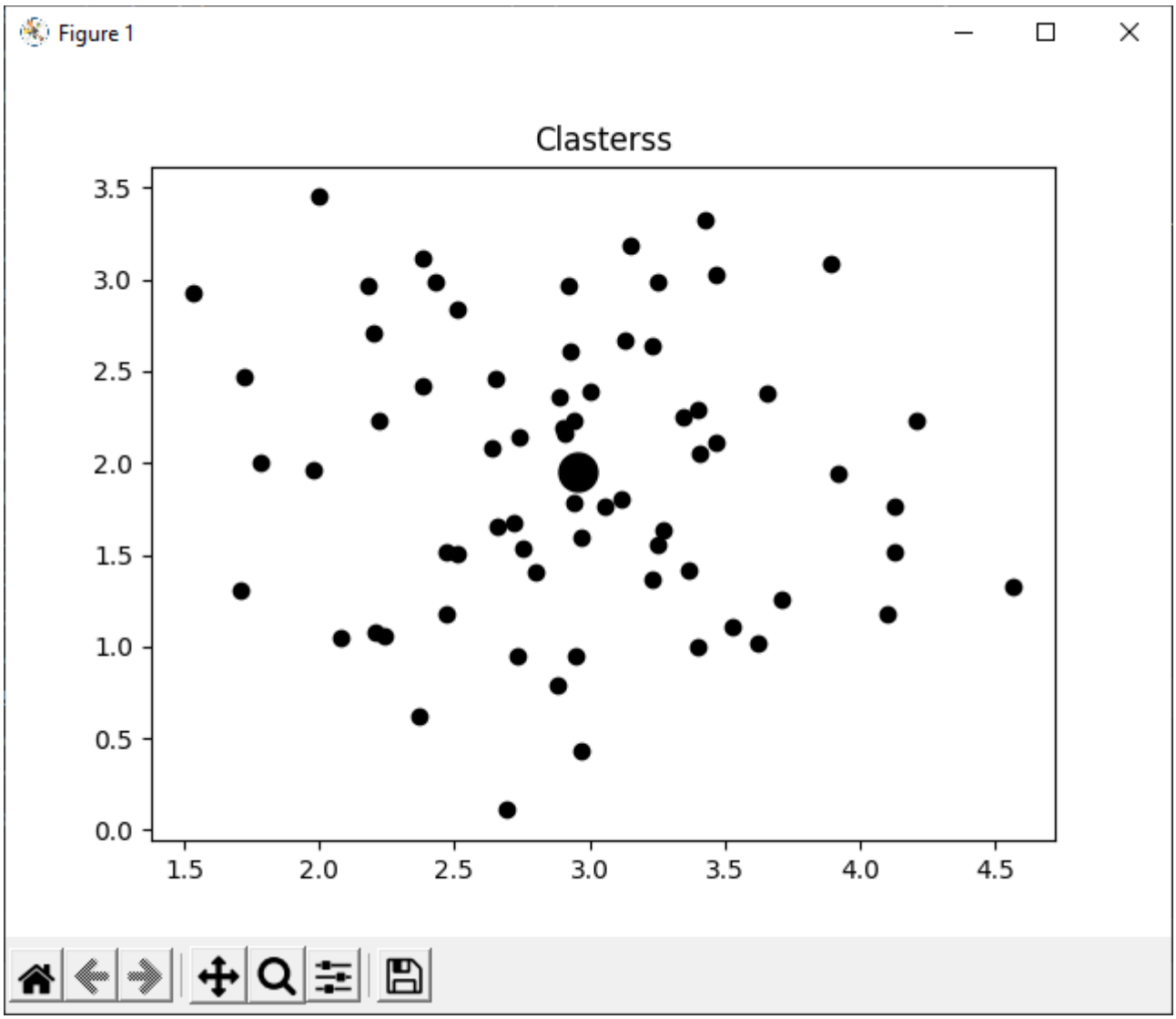

Рисунок 3. Отримана діаграма розсіювання

Після аналізу було отримано наступну діаграму кластерізації вхадних даних. Так як було використано метод зсуву середнього то можна висунути припущення щодо використаного набору. Наприклад що кількість «піків» близька до кількості кластерів.

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

Для вирішення цього завдання було переписано код що було наведено в методичних рекомендаціях через те, що запропонований у ньому функціонал на разі не підтримується в тому вигляді в якому він був описаний. Також через відсутність файлу symbols_mapping.json дані про компанії були прописані напряму в коді програми.

Лістинг програми:

```
#TAsk 4----

import datetime
import json
import numpy as np
import matplotlib.pyplot as plt
from sklearn import covariance, cluster
import yfinance as yf
from yahoofinancials import YahooFinancials



symbols = ["PLUG", "AAPL", "PFE", "JNJ"]
names = ["Plug Power Inc", "Apple", "Pfizer Inc", "Johnson & Johnson"]

openList = []
closeList = []


for symbol in symbols:
    info = yf.Ticker(symbol)
    start_date = datetime.datetime(2003, 7, 3)
    end_date = datetime.datetime(2007, 5, 4)
    quotes = info.history(start=start_date, end=end_date)
    openList.append(quotes.Open)
    closeList.append(quotes.Close)

opening_quotes = np.array(openList).astype(np.float_)
closing_quotes = np.array(closeList).astype(np.float_)

quotes_diff = closing_quotes - opening_quotes

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

edge_model = covariance.GraphicalLassoCV()

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[i]))
```

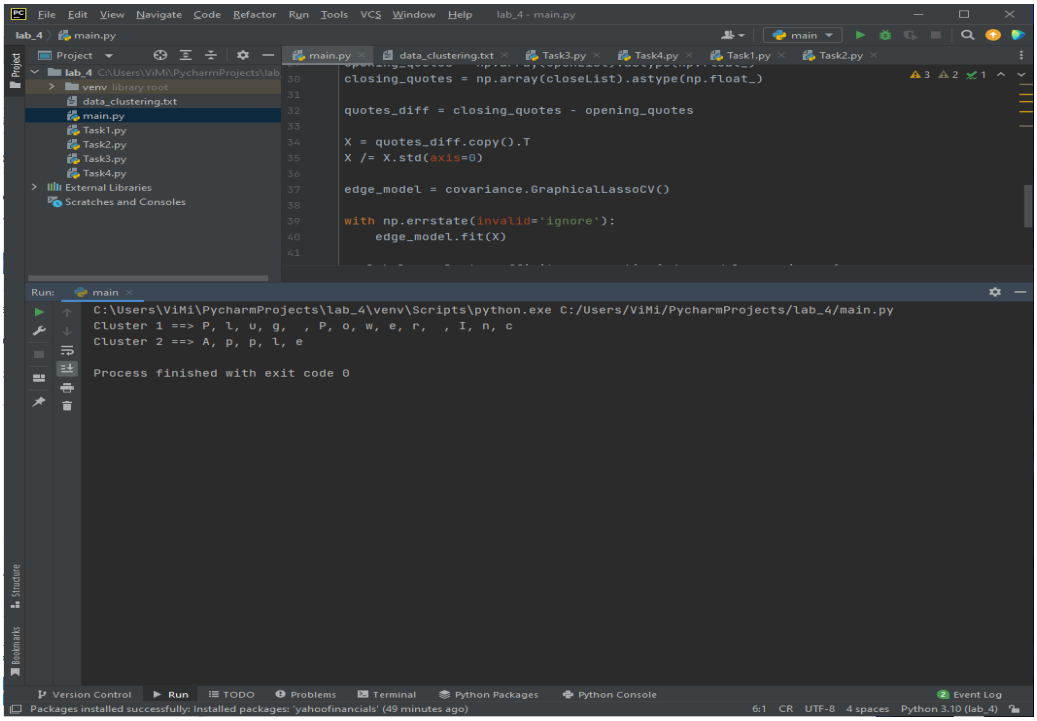

Рисунок 4. Отримані результат

Для поточної конфігурації з отриманого часового проміжку було отримано два великі кластери, Де компанія Plug потрапила в кластер 1, а інші в кластер 2.
Це можна побачити якщо вивести масив кластерів й побачити мітки [0, 1, 1, 1]

Висновок: Було проведено дослідження методів неконтрольованого навчання та засвоєно базові навички кластуріщації данних за подібністю.