Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

ЗВІТ

Про виконання лабораторної роботи №1

Статистичний та кластерний аналіз

Виконав:

Студент групи ФЕП-31

Сворень Ярослав

Перевірив:

Ас. Рибак А. В.

Львів 2022

Мета роботи: Виконати кластерний аналіз для встановлення зв'язку між об'єктами даних у декартовому просторі (x, y) та провести статистичний аналіз для кожного сформованого кластеру.

Хід Роботи:

1. **П**ерш за все, необхідно створити Virtual Environment та інсталювати у нього бібліотеки **Scikit Learn**, **Statistics**. Для цього створюємо директорію для нашої програми, відкриваємо термінал, та вводимо команду **python -m venv venv**, щоб створити папку з віртуальним середовищем під назвою **venv** всередині нашої папки проекту. Далі активуємо наше середовище командою, запустивши файл **venv\Scripts\activate.bat** у командному рядку(або ж **venv\Scripts\Activate.ps1** у **powershell)**. Після цього, встановлюємо наші дві бібліотеки та виводимо повний перелік командою **pip freeze**, який при бажанні можна скопіювати у текстовий файл, назвавши його **requirements.txt**. Щоб покинути віртуальне середовище існує команда **deactivate**.

```
D:\Programs\ZMN\Lab1ZMN>python -m venv venv

D:\Programs\ZMN\Lab1ZMN>venv\Scripts\activate.bat

(venv) D:\Programs\ZMN\Lab1ZMN>pip install scikit-learn
Collecting scikit-learn
```

```
(venv) D:\Programs\ZMN\Lab1ZMN>pip install statistics
```

```
(venv) D:\Programs\ZMN\Lab1ZMN>pip freeze
docutils==0.18.1
joblib==1.1.0
numpy==1.22.3
scikit-learn==1.0.2
scipy==1.8.0
statistics==1.0.3.5
threadpoolctl==3.1.0
```

2. **П**еред складанням програми встановлюємо у віртуальне середовище бібліотеку **matplotlib**, щоб мати можливість візуалізувати вміст наших майбутніх кластерів. Далі складаємо саму програму. Вже з цього моменту звертаємо увагу, що VS Code не підсвічує вміст нашого коду, через віртуальне середовище. Підключаємо нашу бібліотеки та описуємо змінні для кількості елементів та поділу кластерів. Генеруємо наш масив випадкових значень на проміжку значень [0,num_of_elements], після чого копіюємо його вміст на масиви двох осей та малюємо початковий графік.

```
(venv) D:\Programs\ZMN\Lab1ZMN>pip install matplotlib
```

Lab1ZMN.py > ...

```
1   import sklearn.cluster as cl
2   import statistics as st
3   import random as rd
4   from matplotlib import pyplot as plt
5
6   num_of_elements = 50
7   num_of_clusters = 6
```

```
20  array = []
21  x_array = []
22  y_array = []
23  for i in range(0,num_of_elements):
24      array.append([rd.randint(0,num_of_elements),rd.randint(0,num_of_elements)])
25  print("array:\n",array,"\n")
```

```
27  for i in range(0,num_of_elements):
28      x_array.append(array[i][0])
29      y_array.append(array[i][1])
30  plt.scatter(x_array,y_array)
31  plt.title("Original")
32  plt.show()
```

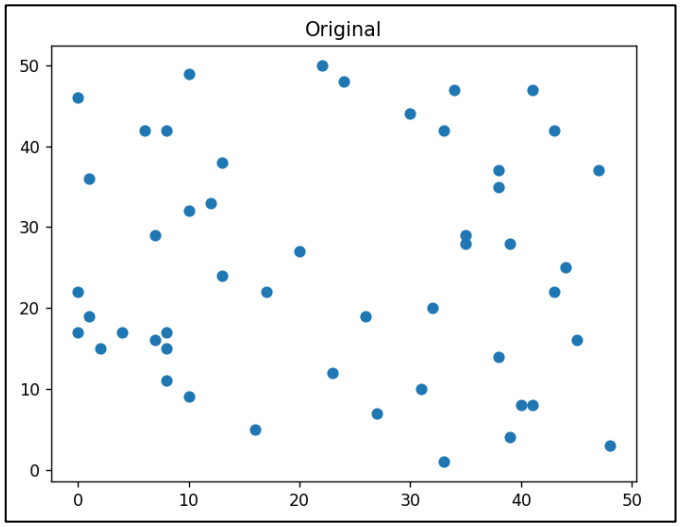
3. Далі за планом йде кластерний аналіз нашого масиву. Згідно з завданням, використовуємо функцію sklearn.cluster.Kmeans, щоб поділити вміст нашого масиву на **num_of_clusters** кластерів, і записуємо результат у **cl_array**. Далі формуємо масиви **cl_labels**, який містить індекси до якого кластеру належить кожен елемент масиву **cl_array** та **cl_centers**, який містить координати центрів кожного з сформованих кластерів.

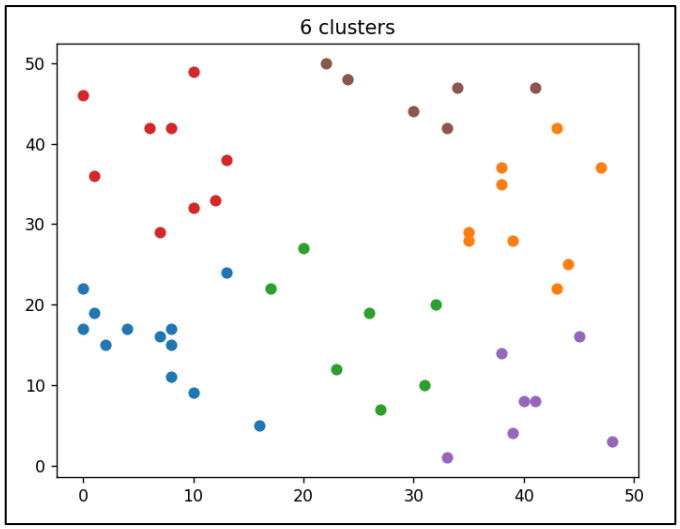
```
34  cl_array = cl.KMeans(n_clusters=num_of_clusters,random_state=0).fit(array)
35  cl_labels = cl_array.labels_
36  cl_centers = cl_array.cluster_centers_
37  print("cluster_labels:\n",cl_labels,"\n")
38  print("cluster_centers:\n",cl_centers)
```

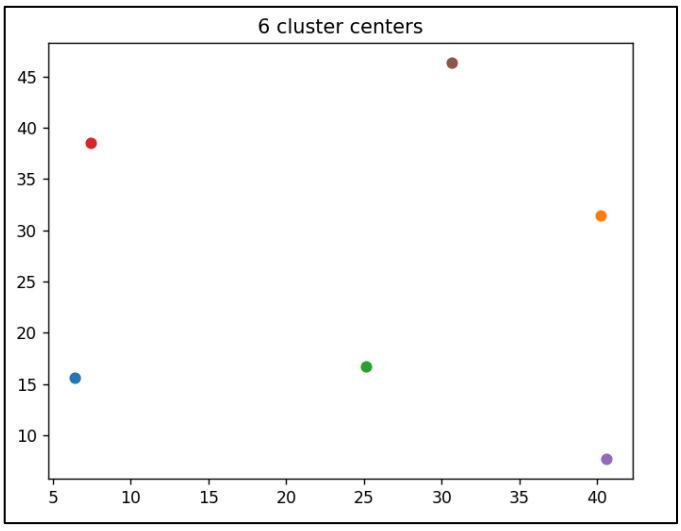
4. **Н**априкінці у нас статистичний аналіз. Для нього ми в циклі знаходимо основну статистику кожної з осей кожного з кластерів, а також функцією **pyplot.scatter()** в циклі малюємо точки кластерів, в результаті чого отримаємо ми зможемо переконатися, що функція **KMeans()** і справді добре поділила елементи нашого масиву.

```
40  for i in range(0,num_of_clusters):
41      temp = []
42      temp2 = []
43      temp3 = []
44      for w in range(0,len(cl_labels)):
45          if(cl_labels[w]==i):
46              temp.append(array[w])
47              temp2.append(array[w][0])
48              temp3.append(array[w][1])
49      print("\nCluster"+str(i)+":\n",temp)
50      print("X values:",temp2)
51      print("Y values:",temp3)
52      print("Min X:",min(temp2))
53      print("Max X:",max(temp2))
54      print("Mean X:",st.mean(temp2))
55      print("Median X:",st.median(temp2))
56      print("Most common X:",*most_common(temp2))
57      print("Min Y:",min(temp3))
58      print("Max Y:",max(temp3))
59      print("Mean Y:",st.mean(temp3))
60      print("Median Y:",st.median(temp3))
61      print("Most common Y:",*most_common(temp3))
62      plt.scatter(temp2,temp3)
63  plt.title(str(num_of_clusters)+" clusters")
64  plt.show()
```

5. Окрім малювання усіх точок масиву є ще можливість намалювати графік центрів кластерів, зберігаючи хронологічні кольори попереднього графіку. Врешті-решт, демонструю вивід програми.







Вся базова статистика, згідно з завданням, була знайдена бібліотекою **Statistics**, окрім функції **mode()**, натомість якої я написав власну, оскільки якщо у масиві є більшого одного числа, яке зустрічається частіше всього у масиві, то воно виводить найближче до початку масиву. Моя ж функція виводить усі елементи, які частіше всього зустрічаються у масиві.

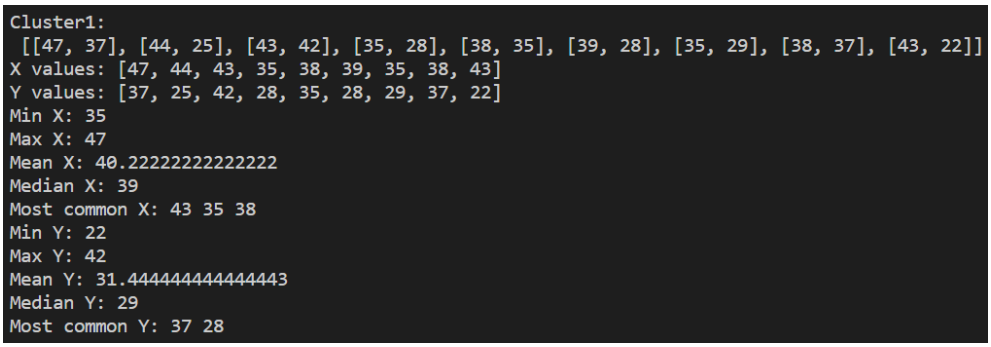
```
 9   def most_common(array:list):
10       temp = []
11       counter = 0
12       for i in range(0,len(array)):
13           if(array.count(array[i])>counter):
14               counter = array.count(array[i])
15       for i in range(0,len(array)):
16           if(array.count(array[i])==counter and array[i] not in temp):
17               temp.append(array[i])
18       return temp
```

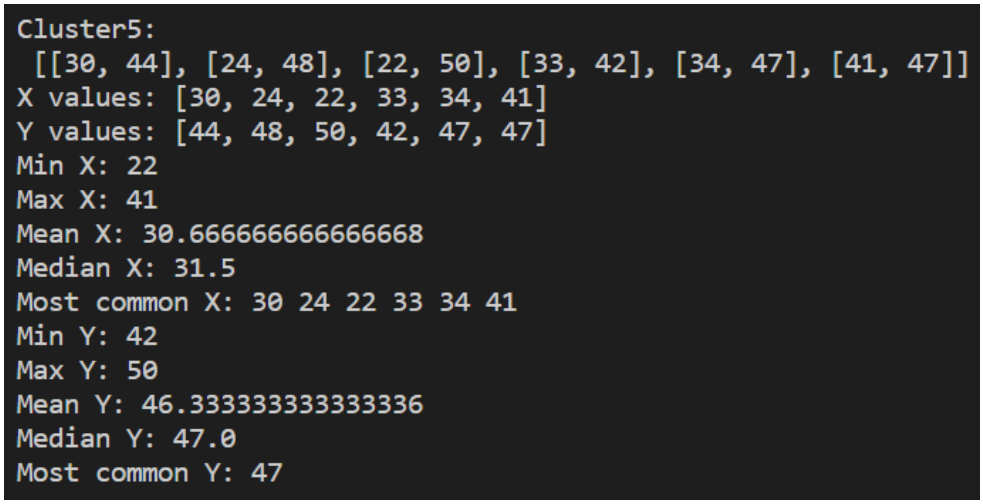
```
(venv) D:\Programs\ZMN\Lab1ZMN>python Lab1ZMN.py
array:
 [[47, 37], [4, 17], [20, 27], [44, 25], [6, 42], [43, 42], [30, 44], [48, 3],
4, 48], [12, 33], [0, 17], [33, 1], [39, 28], [17, 22], [13, 38], [32, 20], [1,
 [23, 12], [8, 15], [10, 9], [2, 15], [16, 5], [35, 29], [45, 16], [10, 32], [3
], [39, 4], [26, 19], [33, 42], [7, 29], [38, 14], [34, 47], [41, 8], [41, 47],

cluster_labels:
 [1 0 2 1 3 1 5 4 1 0 1 2 5 3 0 4 1 2 3 2 0 5 3 4 3 2 0 0 0 0 1 4 3 1 0 0 3
 0 4 2 5 3 4 5 4 5 2 3 1 0]

cluster_centers:
 [[ 6.41666667 15.58333333]
 [40.22222222 31.44444444]
 [25.14285714 16.71428571]
 [ 7.44444444 38.55555556]
 [40.57142857  7.71428571]
 [30.66666667 46.33333333]]
```

```
Cluster0:
 [[4, 17], [13, 24], [0, 17], [1, 19], [8, 15], [10, 9], [2, 15], [16, 5], [8, 17], [0, 22], [8, 11], [7, 16]]
X values: [4, 13, 0, 1, 8, 10, 2, 16, 8, 0, 8, 7]
Y values: [17, 24, 17, 19, 15, 9, 15, 5, 17, 22, 11, 16]
Min X: 0
Max X: 16
Mean X: 6.416666666666667
Median X: 7.5
Most common X: 8
Min Y: 5
Max Y: 24
Mean Y: 15.583333333333334
Median Y: 16.5
Most common Y: 17
```





Висновок: на цій лабораторній роботі було розглянуто процес створення та використання віртуального середовища venv для мови програмування Python, згенеровано масив випадкових чисел та проведено над ним кластерний аналіз, а також статистичний аналіз сформованих кластерів. Для візуалізації було намальовано графік елементів початкового масиву, графік, який показував до якого з кластерів належить кожна з точок сформованих кластерів, та графік центрів сформованих кластерів. В результаті було отримано вміння формування базового віртуального середовища для програми на python, і розглянуто, як підходом кластерного та статистичного аналізу можна поділити один великий масив на необхідну кількість кластерів, після чого буде легше знайти правильний підхід до кожного з них.

Такий підхід може бути доречним, для прикладу, при сортуванні одягу для прання. Нехай є масив одягу, який потрібно поділити за ознаками кольору, температури для прання, чутливості тканні, цей процес можна назвати кластеризацію. Також потрібно підвести статистичний аналіз, порахувавши кількість одягу, об'єм одягу пральної машини та час, за який можна випрати кожен з сформованих кластерів.