МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

**КОМП’ЮТЕРНИЙ ПРАКТИКУМ 1.**

КЛАСТЕРИЗАЦІЇ K-MEANS ДЛЯ АНАЛІЗУ ДАНИХ У PYTHON

Підготував: студент групи ПДМ-51

Гапей Максим Юрійович

Перевірив: викладач

Садовенко Володимир Сергійович

Київ 2024

**Мета роботи:** У частині 1 потрібно виконати кластеризацію клієнтів банку для надання позики різного типу на основі даних про цих клієнтів, надати відповіді на поставлені питання у ході виконання програмного коду. У частині 2 потрібно реалізувати алгоритми кластеризації k-means та k-means++ для ініціалізації центроїдів у Python, а також побудувати криву ліктя, щоб визначити, яка повинна бути потрібна кількість кластерів для вказаного набору даних.

**Частина 1**

Імпортуйте необхідні бібліотеки:

import pandas as pd

import numpy as np

import random as rd

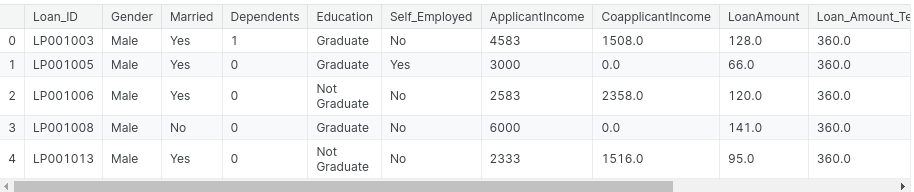
import matplotlib.pyplot as plt

Прочитайте файл CSV і розгляньте перші п’ять рядків даних файлу:

data = pd.read\_csv('clustering.csv')

data.data()

*Який результат ви отримали?*

**

Виберіть ці дві змінні та візуалізуйте точки даних:

X = data[["LoanAmount","ApplicantIncome"]]

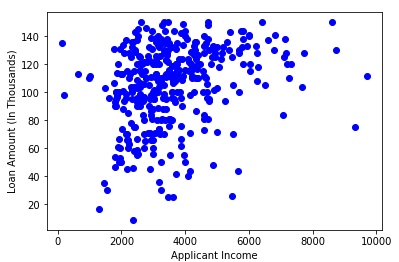
plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')

plt.xlabel('AnnualIncome')

plt.ylabel('Loan Amount (In Thousands)')

plt.show()

*Який результат ви отримали?*



Кроки 1 і 2 k-means – виберіть кількість кластерів (k=3) і виберіть випадковий центроїд для кожного кластера.

K=3

Centroids = (X.sample(n=K))

plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')

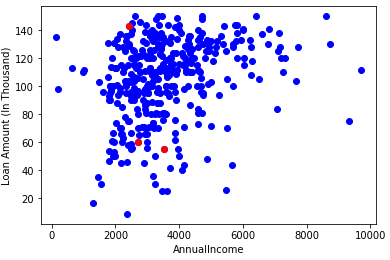
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')

plt.xlabel('AnnualIncome')

plt.ylabel('Loan Amount (In Thousands)')

plt.show()

*Який результат ви отримали?*



Визначте умови реалізації алгоритму кластеризації:

Крок 3 - Призначте всі точки найближчому центроїду кластера.

Крок 4 - Обчисліть центроїди новоутворених кластерів.

Крок 5 - Повторіть кроки 3 і 4.

diff = 1

j=0

while(diff!=0):

XD=X

i=1

for index1, row\_c in Centroids.iterrows():

ED=[]

for index2, row\_d in XD.iterrows():

d1 = (row\_c["ApplicantIncome"]-row\_d["ApplicantIncome"])\*\*2

d2 = (row\_c["LoanAmount"]-row\_d["LoanAmount"])\*\*2

d = sqrt(d1+d2)

ED.append(d)

X[i] = ED

i = i+1

C = []

for index, row in X.iterrows():

min\_dist=row[1]

pos=1

for i in range(K):

if row[i+1] < min\_dist:

min\_dist = row[i+1]

pos = i+1

C.append(pos)

X["Cluster"]=C

Centroids\_new = X.groupby(["Cluster"]).mean()[["LoanAmount", "ApplicantIncome"]]

if j == 0:

diff = 1

j = j+1

else:

diff = (Centroids\_new['LoanAmount'] - Centroids['LoanAmount']).sum() + (Centroids\_new['ApplicantIncome'] - Centroids['ApplicantIncome']).sum()

print(diff.sum())

Centroids = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

*Який результат ви отримали?*

589.4894185619269

609.8777662744718

417.11902675964245

399.19059297247975

215.94836106251185

212.11585654878053

320.86544014225217

235.63273762230665

248.74878646215663

125.16501318300665

52.84741626127729

98.54724443834282

90.64953219227577

18.274686272279013

9.21023994083339

18.345487493007468

46.27013250786139

0.0

Візуалізуйте кластери та наведіть результат виконання програми.

color=['blue','green','cyan']

for k in range(K):

data=X[X["Cluster"]==k+1]

plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])

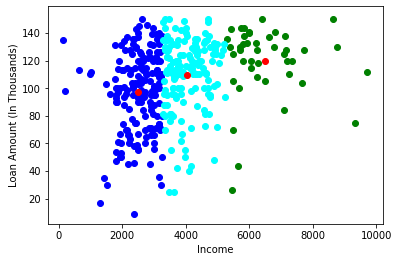
plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')

plt.xlabel('Income')

plt.ylabel('Loan Amount (In Thousands)')

plt.show()

*Поясніть отриманий результат виконання програми.*



**Частина 2**

Імпортуйте необхідні бібліотеки:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

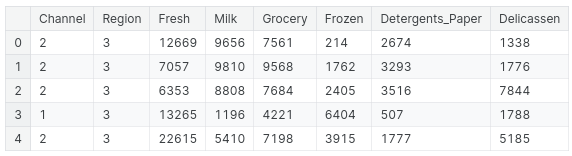
from sklearn.cluster import KMeans

Прочитайте дані про перші п’ять рядків набору даних:

data=pd.read\_csv("Wholesale customers data.csv")

data.head()

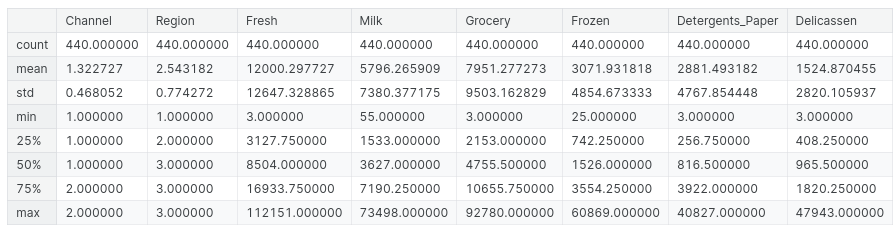
*Який результат ви отримали?*



Проаналізуйте статистичні дані:

data.describe()

*Який результат ви отримали?*



Приведіть усі змінні до однакової величини:

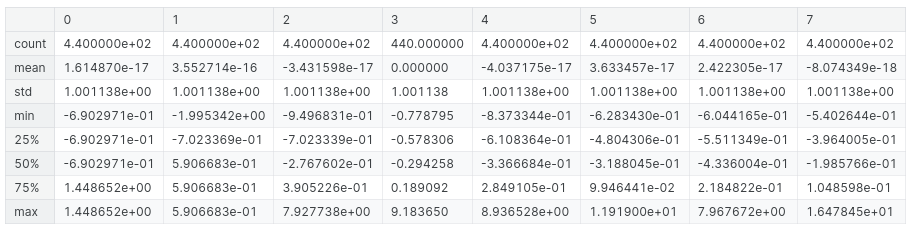
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

data\_scaled = scaler.fit\_transform(data)

pd.DataFrame(data\_scaled).describe()

*Який результат ви отримали?*



Створіть функцію kmeans і помістіть її в дані:

kmeans = KMeans(n\_clusters=2, init='k-means++')

kmeans.fit(data\_scaled)

Оцініть, наскільки добре сформовані кластери. Для цього обчисліть інерцію кластерів:

kmeans.inertia\_

*Який результат ви отримали?*

2599.3855593561393

Спочатку підберіть декілька моделей k-means, і потім в кожній наступній

моделі збільшіть кількість кластерів. Збережіть значення інерції кожної моделі, а потім побудуйте його для візуалізації результату:

SSE = []

for cluster in range(1,20):

kmeans = KMeans(n\_jobs = -1, n\_clusters = cluster, init='k-means++')

kmeans.fit(data\_scaled)

SSE.append(kmeans.inertia\_)

frame = pd.DataFrame({'Cluster':range(1,20), 'SSE':SSE})

plt.figure(figsize=(12,6))

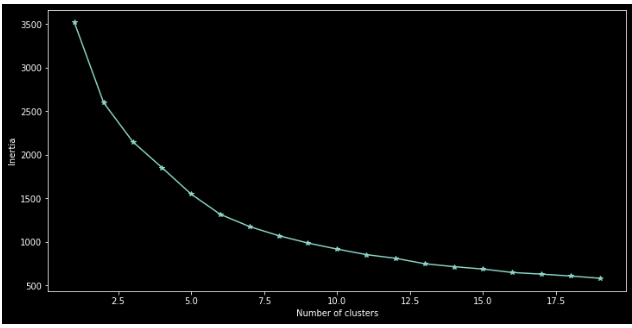
plt.plot(frame['Cluster'], frame['SSE'], marker='o')

plt.xlabel('Number of clusters')

plt.ylabel('Inertia')

*Який результат ви отримали?*

Text(0, 0.5, 'Inertia')



Визначте оптимальне значення кластера за отриманим графіком. Дивлячись на криву ліктя, виберіть число кластерів (від 5 до 8). Встановіть кількість кластерів та підберіть модель:

kmeans = KMeans(n\_jobs = -1, n\_clusters = 5, init='k-means++')

kmeans.fit(data\_scaled)

pred = kmeans.predict(data\_scaled)

Визначте кількість точок у кожному із сформованих вище кластерів:

frame = pd.DataFrame(data\_scaled)

frame['cluster'] = pred

frame['cluster'].value\_counts()

*Який результат виконання програми? Скільки точок потрапило до кожного кластеру?*

2 210

1 125

0 91

4 12

3 2

Name: cluster, dtype: int64

**Висновки:** Під час виконання практичної роботи, було проведено кластеризацію клієнтів та розроблено аглоритм кластеризації.