Въведение в Машинното обучение
Лабораторно упражнение №10

Клъстеризация

1. Цел на упражнението.

Целта на лабораторното упражнение е студентите да усвоят темата клъстеризация, като я приложат при решаване на задачи за клъстеризация.

2. Алгоритъм в Python използван в пример 1 и задачи 1, 2, 3 и 4

а) Алгоритъм на к-средния (k means)

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

3. Допълнителни пояснения към кода на пример 1

а) за ред 5

5) np.random.seed(0)

np.random.seed(0) е функция в NumPy, която задава сийд (начално число) на генератора на случайни числа в NumPy.

Сийдът е цяло число, което се използва за инициализиране на генератора на случайни числа. Когато се зададе конкретно сийд число, генераторът на случайни числа ще генерира последователност от случайни числа, която винаги е еднаква за дадено сийд число. Това означава, че при повторно изпълнение на кода със същия сийд, ще получите едни и същи случайни числа.

В случая, функцията np.random.seed(0) е използвана, за да се генерират едни и същи случайни числа при всеки стартиране на програмата, което е полезно за дебъгване и тестване на кода, защото така ще можете да репродуцирате резултатите от изпълнението на кода.

б) за ред 6

6) X = np.random.randn(50, 2)

Кодът X = np.random.randn(50, 2) означава, че създавате масив от 50 точки в двумерно пространство.

Броят на колоните в масива е два, което означава, че всеки ред от масива представлява две координати (x и y) на точката в двумерното пространство.

в) за ред 8

8) kmeans = KMeans(n_clusters=2)

kmeans = KMeans(n_clusters=2) създава обект от класа KMeans от модула sklearn.cluster, който използва алгоритъма за клъстеризация "K-means".

n_clusters=2 задава броят на клъстерите, които алгоритъмът трябва да открие.

г) за ред 9

9) kmeans.fit(X)

при извикване на метода fit() върху обекта kmeans, алгоритъмът ще групира всички точки от множеството X в 2 клъстера, като ще се опита да минимизира разстоянието между точките в един клъстер и да го максимизира между различните клъстери.

д) за ред 11

11) plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_)

Кода plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_) визуализира точките, които са създадени в X и разделя на клъстери чрез алгоритъма за клъстеризация KMeans.

X[:, 0] взема всички стойности от първата колона на масива X, която съдържа x координатите на точките, и ги подава като първи аргумент на plt.scatter(). Аналогично X[:, 1] взема всички стойности от втората колона на масива X, която съдържа y координатите на точките, и ги подава като втори аргумент на plt.scatter().

c=kmeans.labels_ указва на plt.scatter() да използва клъстерните маркировки, които са изчислени от алгоритъма за клъстеризация KMeans в променливата kmeans.labels_. Тези маркировки указват на plt.scatter() да визуализира точките от всеки клъстер в различен цвят.

е) за ред 12

12) plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='x', s=100, linewidths=3, color='r')

plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='x', s=100, linewidths=3, color='r') рисува центровете на клъстерите върху графиката.

Конкретно kmeans.cluster_centers_ връща координатите на центровете на клъстерите, намирани от алгоритъма за клъстеризация "K-means". От тези координати се взимат само първата ([:, 0]) и втората ([:, 1]) колона, които съответстват на координатите на оста x и y, съответно.

marker='x' определя маркера за центровете на клъстерите, който в този случай е кръстче.

s=100 определя размера на маркера. В този случай, размерът е 100 пиксела.

linewidths=3 определя дебелината на линиите на маркера, която е 3 пиксела.

color='r' определя цвета на маркера. В този случай, цветът е червен ('r').

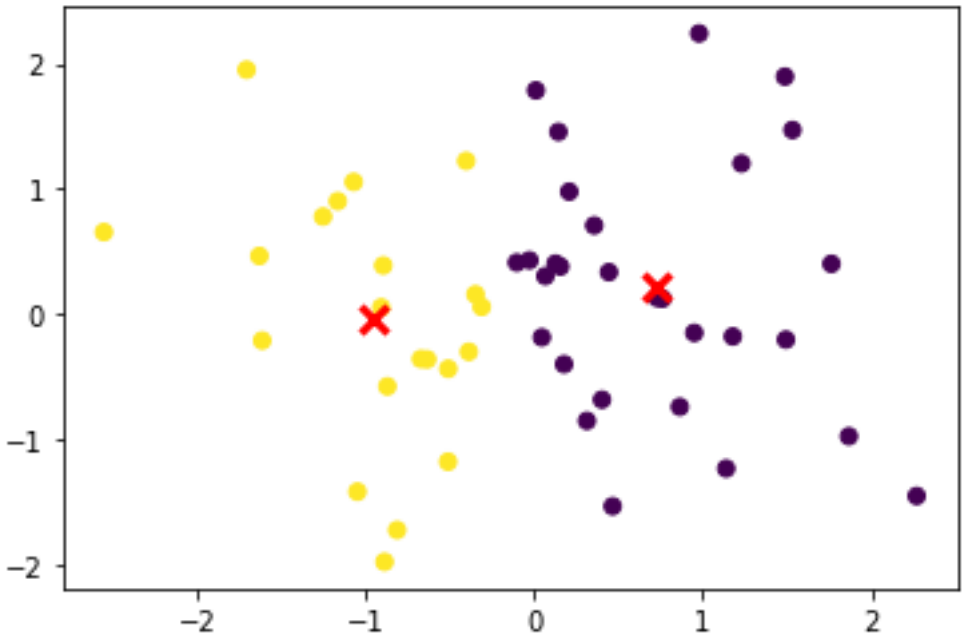
Пример 1. Клъстеризация на точки в двумерното пространство

С помощта на метода KMeans да се раздели на два клъстера множество от 50 точки в двумерно пространство, така че точките в единия клъстер да са максимално близо до

средната точка на този клъстер, а точките в другия клъстер да са максимално близо до средната точка на този клъстер. Да се визуализират резултатите.

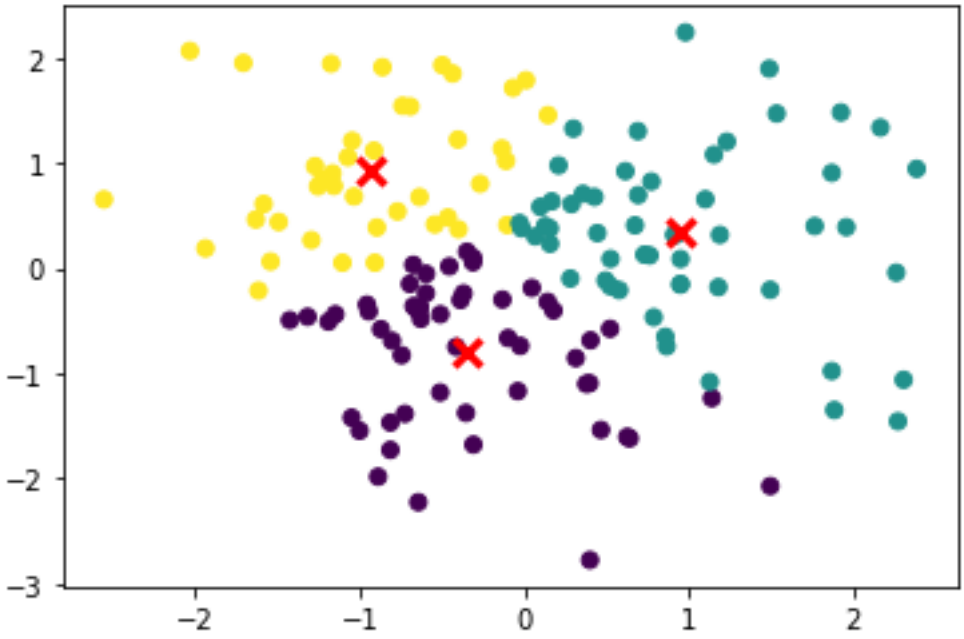
```
1)  import numpy as np
2)  import matplotlib.pyplot as plt
3)  from sklearn.cluster import KMeans

4)  # Генериране на случайни точки
5)  np.random.seed(0)
6)  X = np.random.randn(50, 2)

7)  # Използване на алгоритъм за клъстеризацията на точките в 2 клъстера
8)  kmeans = KMeans(n_clusters=2)
9)  kmeans.fit(X)

10)     # Визуализация на резултатите
11)     plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_)
12)     plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
    marker='x', s=100, linewidths=3, color='r')
13)     plt.show()
```

Резултат:



Задача 1. Клъстеризация на точки в двумерното пространство

С помощта на алгоритъма KMeans да се раздели на три клъстера множество от 150 точки в двумерно пространство. Да се визуализират резултатите. (1 точка)



4. Допълнителни пояснения към кода на задача 2

а) за ред 11

11) kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)

Този ред код създава обект от класа KMeans от библиотеката scikit-learn, който ще се използва за клъстеризация на данните.

На ред 11, данните за атрибутите на признаците се извличат от data и се съхраняват в променливата X.

n_clusters: това е цялото число, което указва броят на клъстерите, който искаме да създадем. В кода, това число е настроено да се променя във всеки един цикъл на базата на стойността на променливата i, която също е дефинирана в кода и се движи в интервала [1, 10].

init: това е методът, който ще се използва за инициализация на центровете на клъстерите. В този случай е избран методът "k-means++".

'k-means++' е метод за инициализация на центровете на клъстерите, който се използва в алгоритъма за клъстеризация K-means. Този метод е по-ефективен и по-стабилен от случайната инициализация, която е стандартната практика в K-means.

При инициализацията с 'k-means++' първият център на клъстера се избира произволно от множеството на данните. След това, за всяка следваща итерация, центърът на клъстера се избира като се избере нов център, който е далечен от всички вече избрани центрове. Това увеличава вероятността да се получи по-добра клъстеризация и помага да се избегнат неравномерни клъстери или съвпадащи клъстери.

max_iter: това е максималният брой итерации, които KMeans алгоритъмът може да извърши, преди да прекрати обучението.

n_init: това е броят на различните начални точки, които ще се използват при инициализацията на центровете на клъстерите. Този параметър определя колко пъти ще се извършва KMeans алгоритъма с различни начални точки и ще се избере този с най-ниска WCSS.

random_state: този параметър контролира генератора на случайни числа, използван при инициализацията на KMeans модела. Когато той се установи на дадено цяло число, генерираните случайни числа ще бъдат предсказуеми и повторяеми при всяко изпълнение на програмата със същото число. Това помага да се получат по-предсказуеми резултати, което е важно при използването на случайни генератори в научните изчисления.

б) за ред 13

13) wcss.append(kmeans.inertia_)

WCSS е кратко за "Within-Cluster-Sum-of-Squares", което е метрика за оценка на качеството на клъстеризацията. По-конкретно, това е сумата на разстоянията на всеки елемент в клъстера до неговия център.

В този ред на кода се добавя стойността на WCSS (Within-Cluster Sum of Squares) към списъка wcss. WCSS е мярка за вариацията вътре в клъстерите и се изчислява като сумата от квадратните разстояния между точките и центровете на техните клъстери.

Методът kmeans.inertia_ връща текущата стойност на WCSS след обучението на KMeans модела с текущия брой клъстери, който е зададен в момента на създаването му в ред 11.

Функцията append() добавя стойността на kmeans.inertia_ към края на списъка wcss, който съхранява WCSS стойностите за всяка итерация на KMeans модела с различен брой клъстери. Това се извършва в цикъла на редове 10-13, който променя броя на клъстерите от 1 до 10, така че списъкът wcss ще съдържа WCSS за всеки един от тези 10 възможни броя на клъстерите.

В крайна сметка, този списък wcss може да бъде използван за визуализиране на "лакатовата крива" (Elbow Curve), която помага за определяне на оптималния брой на клъстери в модела.

в) ред 21

21) kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)

Ред 21 създава обекта kmeans от тип KMeans. Този обект е имплементация на алгоритъма за клъстеризация "K-means", който се използва за групиране на данни по определен брой клъстери, като се максимизира подобието между данните в един клъстер, но се минимизира подобието между различните клъстери.
Аргументите на KMeans() конструктора задават настройките на алгоритъма. В този случай:

n_clusters=3 задава броя на клъстерите, който kmeans ще опита да създаде. Тук са избрани три клъстера.
init='k-means++' задава начина на инициализация на центровете на клъстерите.
'k-means++' е метод за инициализация, който цели да постави началните центрове на клъстерите така, че да има по-голяма вероятност да се постигне по-добра клъстеризация.
max_iter=300 задава максималния брой итерации, които алгоритъмът ще извърши, преди да се спре. В този случай са зададени 300 итерации.
n_init=10 задава броя на различните начални позиции на центровете, които ще се използват за създаване на клъстерите. Тук са избрани 10 начални позиции, за да се избегне локалното минимум при оптимизацията.
random_state=0 задава началната стойност на генератора на случайни числа. Това прави възпроизвеждането на резултатите предвидимо, тъй като генераторът на случайни числа ще върне същите стойности, ако се зададе същата стойност на random_state.
Така, обектът kmeans е създаден с настройките, които са избрани в ред 21, и може да се използва за клъстеризация на данните, които са предоставени в X.

в) ред 22

22) pred_y = kmeans.fit_predict(X)

Методът fit_predict() на класа KMeans извършва клъстеризацията на данните в матрицата X и връща масив, който съдържа номерата на клъстерите, в които всяка точка от X е присвоена. Това означава, че pred_y ще бъде масив с дължина, равен на броя на точките в X, като всяка стойност в масива ще бъде номерът на клъстера, към който е присвоена съответната точка.

г) ред 24

24) plt.scatter(X[:,0], X[:,1], c=pred_y)

Методът scatter() на обекта plt рисува точки в двумерната равнина, като координатите на точките в равнината са взети от X[:,0] и X[:,1], които са съответно първата и втората колона от матрицата X. Тези колони се използват като координати на точките в графиката.
Аргументът c на метода scatter() указва, че точките трябва да бъдат оцветени в зависимост от присвоените им клъстери в pred_y. Така всички точки в един и същи клъстер ще бъдат оцветени в един и същи цвят. Конкретните цветове на клъстерите се определят автоматично от Matplotlib.

г) ред 25

25) plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red')

Параметрите на функцията plt.scatter() са следните:

kmeans.cluster_centers_[:, 0] и kmeans.cluster_centers_[:, 1] - това са координатите на центровете на клъстерите, които са изчислени от метода k-means clustering. [:, 0] и [:, 1] съответно извличат x и y координатите на всички центрове на клъстери.

s=300 - определя размера на точките в пиксели. Тук е избран размер 300.
c='red' - определя цвета на точките. Тук е избрано червено ('red').
Този ред код създава графика, на която центровете на клъстерите се изобразяват като големи червени точки.

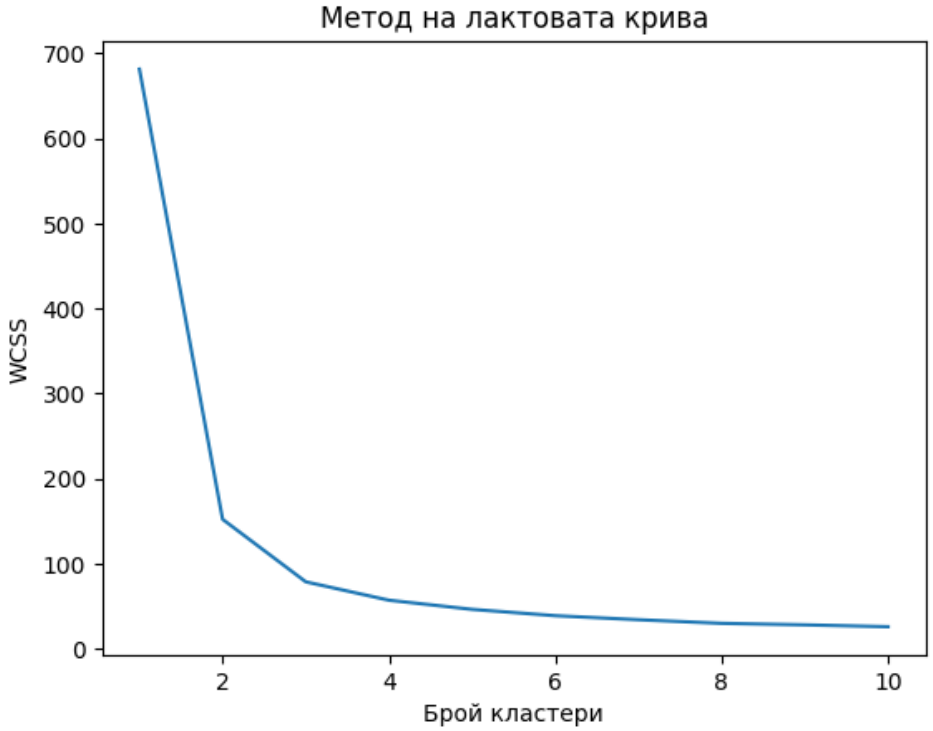
Задача 2. Клъстеризация на данни от Iris цветен набор с помощта на алгоритъма KMeans

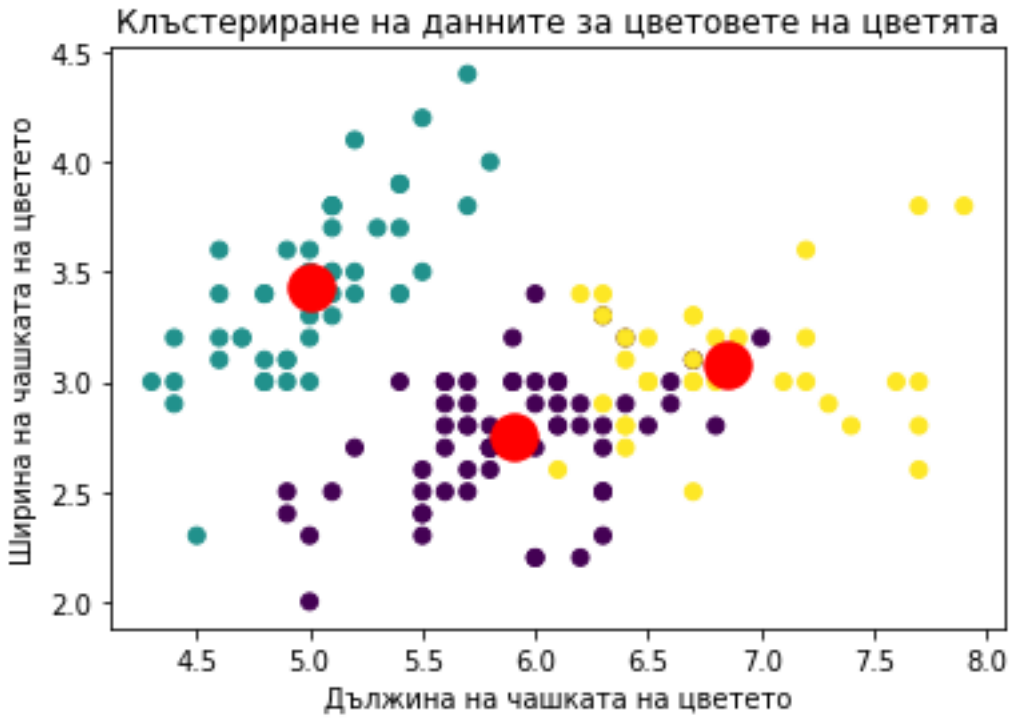
С помощта на метода на лактовата крива, да се определи оптималния брой клъстери за клъстеризиране на данните от Iris цветен набор. Да се визуализира лактовата крива, която показва вариацията на обобщаващата сума на квадратите на отклоненията (WCSS) за всяко количество клъстери. Да се клъстеризират данните с алгоритъма KMeans, като се използва оптималния брой клъстери, определен от лактовата крива. Да се визуализира клъстеризацията на Iris цветния набор, като се изобразят различните клъстери с различни цветове както и центровете на клъстерите. В програмата на задачата липсват части от кода, които трябва да въведете. Това са редове: 6, 7, от 16 до 19 ред, от 26 до 29 ред (1 точка).

```
1)  from sklearn.datasets import load_iris
2)  from sklearn.cluster import KMeans
3)  import pandas as pd
4)  import matplotlib.pyplot as plt


5)  # Зареждане на данните
6)
7)


8)  # Използване на метода на лактовата крива за определяне на броя клъстери
9)  wcss = []
10)         for i in range(1, 11):
11)         kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10,
    random_state=0)
12)         kmeans.fit(X)
13)         wcss.append(kmeans.inertia_)


14)         # Визуализиране на лактовата крива
15)         plt.plot(range(1, 11), wcss)
16)
17)
18)
19)
```

```
20)     # Клъстеризация на данните с K-средни
21)     kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10,
        random_state=0)
22)     pred_y = kmeans.fit_predict(X)


23)     # Визуализиране на резултата
24)     plt.scatter(X[:,0], X[:,1], c=pred_y)
25)     plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300,
        c='red')
26)
27)
28)
29)
```

Резултати





5. Допълнителни пояснения към кода на задача 3

От ред 12 до ред 16

12) fig, ax = plt.subplots(2, 5, figsize=(8, 3))

13) centers = kmeans.cluster_centers_.reshape(10, 8, 8)

14) for axi, center in zip(ax.flat, centers):

15) axi.set(xticks=[], yticks=[])

16) axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)

Ред 12 създава изображение с 2 реда и 5 колони, използвайки функцията subplots() на библиотеката за визуализация на данни Matplotlib.

figsize=(8, 3) задава размерите на изображението в инчове.

Ред 13 преобразува 1D масива, представляващ центровете на клъстерите, в многомерен масив 10 x 8 x 8, където 10 е броят клъстери, а 8 x 8 е размерът на всяко изображение.

Ред 14 създава цикъл, който обхожда всички аксесоари (subplot) в изображението и всяко центрирано изображение на клъстера.

Ред 15 задава xticks и yticks на празни списъци [], за да се премахнат маркерите на осите.

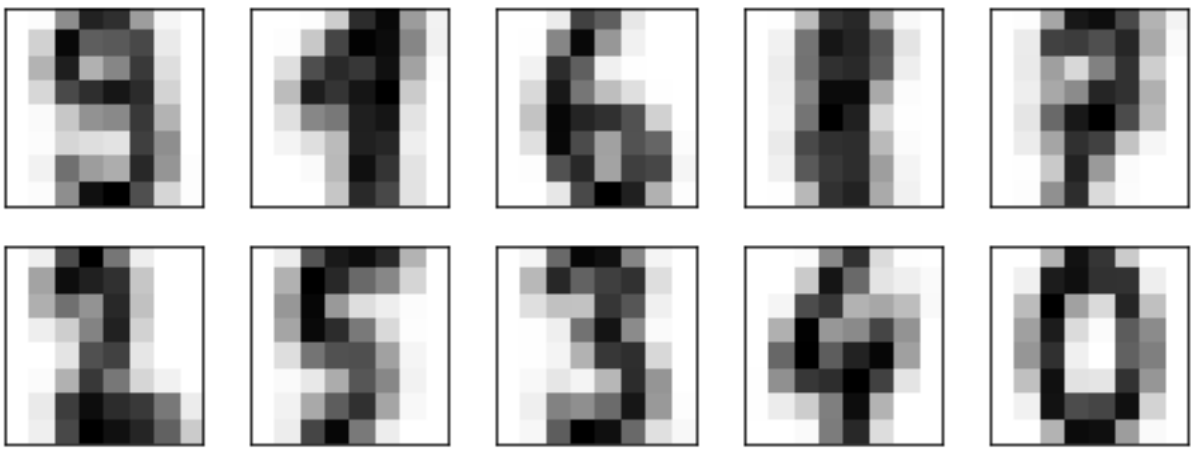
Ред 16 използва метода imshow() на Matplotlib, за да покаже текущото центрирано изображение на клъстера, като се задава cmap=plt.cm.binary за черно-бяла цветова карта.

Задача 3. Клъстеризация на изображения на цифри с помощта на алгоритъма KMeans

Да се клъстеризират изображения на цифри, като се използва библиотеката scikit-learn, за да се зареди набор от данни за цифри, и след това да се използва алгоритъм за клъстеризация (KMeans) за да се групират подобните изображения на цифри в клъстери. Да се визуализират центровете на всеки клъстер, като ги показва като изображения на цифри. В програмата на задачата липсват части от кода, които трябва да въведете. Това са редове: 5, 6, 7, 9 и 10 (1 точка).

```
1)  from sklearn.datasets import load_digits
2)  from sklearn.cluster import KMeans
3)  import matplotlib.pyplot as plt


4)  # Зареждане на данните
5)
6)
7)


8)  # Използване на алгоритъм за клъстеризацията в 10 клъстера
9)
10)


11)     # Показване на центровете на клъстерите
12)     fig, ax = plt.subplots(2, 5, figsize=(8, 3))
13)     centers = kmeans.cluster_centers_.reshape(10, 8, 8)
14)     for axi, center in zip(ax.flat, centers):
15)     axi.set(xticks=[], yticks=[])
16)     axi.imshow(center, interpolation='nearest', cmap=plt.cm.binary)
17)     plt.show()
```

Резултати



6. Допълнителни пояснения към кода на задача 4

а) ред 6

6) data = pd.read_excel(url)

Функцията pd.read_excel() връща обект от тип DataFrame, който представлява таблица с данни, където всеки ред съответства на ред от Excel файла, а колоните съответстват на колоните в таблицата в Excel файла.

б) ред 8

8) data.dropna(inplace=True)

Методът dropna() се използва за премахване на редове или колони от DataFrame, които съдържат липсващи (NaN) стойности. Ако параметърът inplace е зададен като True, DataFrame обектът се модифицира, т.е. методът изтрива редовете или колоните с NaN стойности и променя изходния DataFrame.

Параметърът inplace=True в ред 8 задава методът dropna() за да модифицира DataFrame обекта "data", като изтрива редовете, които съдържат поне една NaN стойност. След изпълнението на този ред, DataFrame обектът "data" вече няма да съдържа липсващи стойности (NaN), ако има такива в оригиналния DataFrame**.**

в) ред 15

15) plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=kmeans.labels_, cmap='viridis')

Този ред от кода използва библиотеката matplotlib в Python за да построи scatter plot (точкова графика) на двата признака (features) "Quantity" и "UnitPrice", които са избрани в DataFrame обекта "X".

Функцията scatter() приема два аргумента, които представляват данните, които трябва да бъдат визуализирани по X и Y осите. В случая X.iloc[:, 0] означава, че трябва да се вземат всички редове от първата колона на DataFrame обекта "X", а X.iloc[:, 1] означава, че трябва да се вземат всички редове от втората колона.

Третият аргумент на функцията scatter() - c=kmeans.labels_ задава цвета на точките в графиката в зависимост от клъстерите, към които са присвоени. Тук kmeans.labels_ е масив, който съдържа етикетите (labels) на клъстерите, към които са присвоени точките от обекта на клъстеризацията.

Последният аргумент cmap='viridis' задава цветовата палитра, която да се използва за да се представят различните клъстери с различни цветове. В този случай се използва вградената палитра "viridis", която отразява промените в цвета като промените в яркостта и използва зелената като основен цвят.

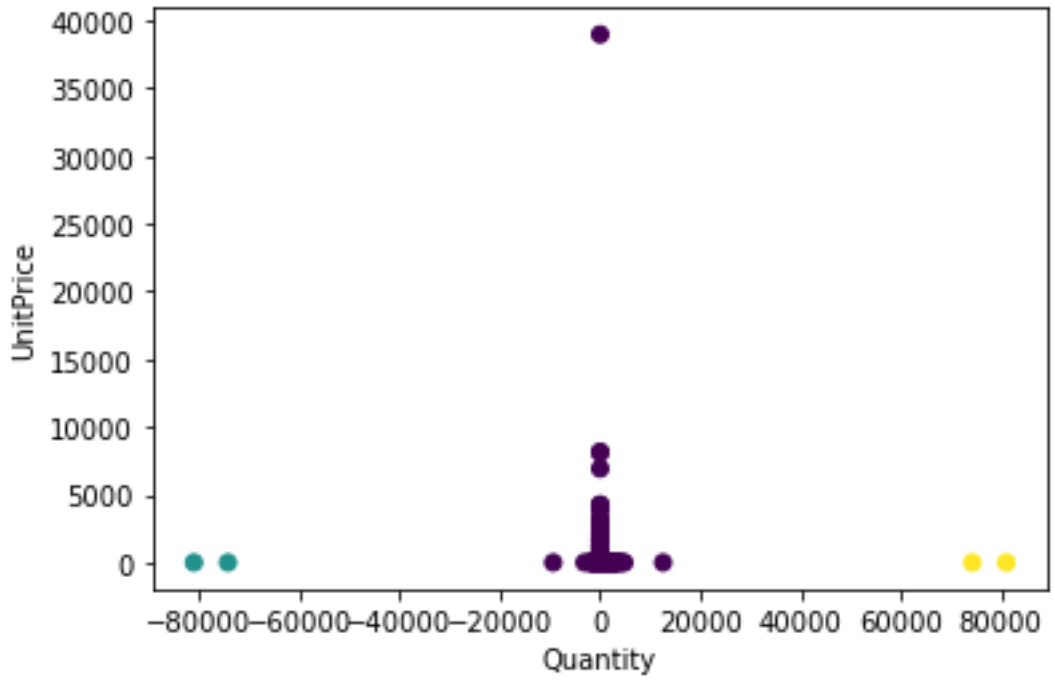
Задача 4. Клъстеризация на данни за онлайн магазин с помощта на алгоритъма KMeans

Въведение в Машинното обучение
Лабораторно упражнение №10

Да се осъществи клъстеризация в 3 клъстера върху данни за онлайн магазин, където имаме информация за количество и цена на продуктите. Да се визуализират резултатите, като визуализацията да показва как се разделят данните в трите клъстера, които сме определили. В програмата на задачата липсват части от кода, които трябва да въведете. Това са редове: 1, 2, 3, 12, 13, 16, 17 и 18 (1 точка).

```
1)
2)
3)


4)  # Зареждаме данните
5)  url = "https://archive.ics.uci.edu/ml/machine-learning-
    databases/00352/Online%20Retail.xlsx"
6)  data = pd.read_excel(url)


7)  # Подготвяме данните, като премахваме празните полета
8)  data.dropna(inplace=True)


9)  # Определяме факторите, по които ще клъстеризираме
10)     X = data[["Quantity", "UnitPrice"]]


11)     # Използване на алгоритъм за клъстеризацията в 3 клъстера
12)
13)


14)     # Визуализираме резултатите
15)     plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=kmeans.labels_, cmap='viridis')
16)
17)
18)
```

Резултати:



7. Алгоритъм в Python използван в пример 2 и задача 5

Алгоритъм с йерархичната клъстеризация - Agglomerative Clustering.

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

8. Допълнителни пояснения към кода на пример 2

а) ред 8

8) s = requests.get(url).content

Този ред от кода използва библиотеката requests в Python за да изпрати GET заявка към уебсайта на зададения URL адрес и да върне съдържанието на отговора като байтов низ.

По-точно, функцията requests.get(url) изпраща GET заявка към уебсайта, зададен от URL адреса в променливата "url", за да заяви информация от този уебсайт. Това може да бъде HTML съдържание, JSON обекти или други формати.

Методът .content връща съдържанието на HTTP отговора, получен след изпращане на GET заявката, като байтов низ. Така че, променливата "s" ще съдържа байтов низ, който представлява съдържанието на отговора на заявката.

б) ред 9

9) dataset = pd.read_csv(io.StringIO(s.decode('utf-8')), header=None)

Този ред код е предназначен за зареждане на CSV файл в Pandas DataFrame от паметта на компютъра.

pd е псевдоним на библиотеката pandas

read_csv() е метод на pandas, който се използва за зареждане на CSV файлове

io.StringIO() създава текстов файл в паметта, който се използва за прочитане на данните от CSV файла

s.decode('utf-8') декодира байтовия обект s от кодиране UTF-8 в низ от символи

header=None указва, че CSV файла няма заглавен ред

С други думи, този ред код зарежда данните от CSV файл в Pandas DataFrame, като данните се прочитат от байтов обект в паметта на компютъра, декодират се от UTF-8 кодиране и няма заглавен ред в CSV файла.

в) ред 10

10) X = dataset.iloc[:, :-1].values

Този ред код присвоява на променливата X стойностите на всички редове и колони, без последната колона, от DataFrame dataset.

iloc е метод на Pandas DataFrame, който се използва за индексиране на DataFrame чрез целочислени позиции.

[:, :-1] индексира всички редове и всички колони, без последната колона.

: указва, че трябва да се вземат всички редове,

а :-1 указва, че трябва да се вземат всички колони без последната.

values връща стойностите на DataFrame като двумерен масив NumPy.

С други думи, този ред код създава двумерен масив X, който съдържа данните за обучение на модела. Тези данни включват всички колони на dataset, без последната. Последната колона се счита за целева променлива и се използва за оценка на точността на модела.

г) ред 12

12) cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')

Този ред код инициализира обект от клас AgglomerativeClustering от библиотеката scikit-learn, който се използва за йерархично клъстериране на данни.

Обяснението на параметрите е следното:

n_clusters=3 указва, че трябва да се формират 3 клъстера. Това е задължителен параметър за AgglomerativeClustering.

affinity='euclidean' указва, че разстоянието между две точки ще бъде измервано с евклидовото разстояние. Това е разстоянието, което се използва за определяне на близостта между точките.

linkage='ward' указва, че методът на обединение, който се използва за сливане на клъстерите, е методът на Уорд. Този метод минимизира разстоянието между точките в новообразуваните клъстери след сливането им.

С други думи, този ред код инициализира обект от клас AgglomerativeClustering с 3 клъстера, евклидово разстояние между точките и метод на Уорд за сливане на клъстерите. Обектът cluster може да бъде използван за клъстериране на данни, като се извиква методът fit_predict() върху данните.

д) ред 15

15) plt.scatter(X[cluster.labels_ == 0, 0], X[cluster.labels_ == 0, 1], s=50, marker='o', color='red', label='Cluster 1')

Този ред код рисува точки на графика, като използва резултатите от клъстеризацията, която е извършена с помощта на обекта cluster от клас AgglomerativeClustering.

X[cluster.labels_ == 0, 0] и X[cluster.labels_ == 0, 1] са координатите на точките, които принадлежат на клъстер 1.

cluster.labels_ връща масив с етикетите на клъстерите, където 0 означава клъстер 1. X[cluster.labels_ == 0] връща всички точки от масива X, които принадлежат на клъстер 1. Последните две индексации [:, 0] и [:, 1] връщат първата и втората колона на масива, съответно.

s=50 задава размера на точките на графиката.

marker='o' задава формата на точките на графиката.

color='red' задава цвета на точките на графиката.

label='Cluster 1' задава легендата на точките на графиката.

С други думи, този ред код рисува точки на графика, които принадлежат на клъстера 1 и са оцветени в червено. Размерът на точките е 50, а формата им е кръг. Тази информация също така е включена в легендата на графиката като "Cluster 1".

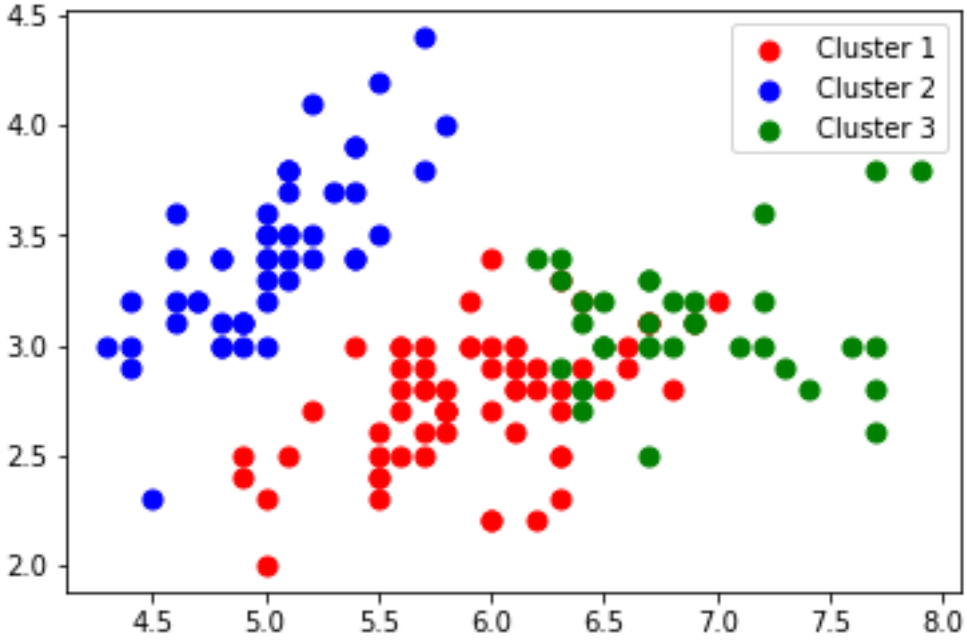
Пример 2. Клъстеризация на данни от Iris цветен набор с помощта на йерархичната клъстеризация

Да се клъстеризират данните от Iris цветен набор с помощта на йерархичната клъстеризация, като данните се заредят от интернет. Да се визуализира клъстеризацията на Iris цветния набор, като се изобразят различните клъстери с различни цветове както и центровете на клъстерите.

1) *import requests*
2) *import io*
3) *import pandas as pd*

```
4) from sklearn.cluster import AgglomerativeClustering
5) import matplotlib.pyplot as plt


6) # изтегляне на данни от интернет и записването им в DataFrame
7) url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
8) s = requests.get(url).content
9) dataset = pd.read_csv(io.StringIO(s.decode('utf-8')), header=None)
10) X = dataset.iloc[:, :-1].values


11) # клъстеризация на данните с използване на йерархичен алгоритъм за групиране
12) cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean',
    linkage='ward')
13) cluster.fit_predict(X)


14) # визуализация на резултатите
15) plt.scatter(X[cluster.labels_ == 0, 0], X[cluster.labels_ == 0, 1], s=50, marker='o',
    color='red', label='Cluster 1')
16) plt.scatter(X[cluster.labels_ == 1, 0], X[cluster.labels_ == 1, 1], s=50, marker='o',
    color='blue', label='Cluster 2')
17) plt.scatter(X[cluster.labels_ == 2, 0], X[cluster.labels_ == 2, 1], s=50, marker='o',
    color='green', label='Cluster 3')
18) plt.legend()
19) plt.show()
```

Резултати:



Задача 5. Клъстеризация на данни от Iris цветен набор с помощта на йерархичната клъстеризация

Да се клъстеризират данните от Iris цветен набор с помощта на йерархичната клъстеризация, като данните се заредят от библиотеката sklearn. Да се визуализира клъстеризацията на Iris цветния набор, като се изобразят различните клъстери с различни цветове както и центровете на клъстерите. (1 точка)