به نام خدا

تمرین ۳

Clustering

درس مبانی داده کاوی

استاد : دکتر محسن غلامی

کسری صمدی<۹۹۳۶۲۳۰۳۰>

بهمن ۱۴۰۲

فهرست

فهرست تصاویر

مقدمه

در علم داده، خوشه‌بندی (clustering) به مجموعه‌ای از الگوریتم‌هایی گفته می‌شود که با توجه به ویژگی‌های مشترک بین داده‌ها، آن‌ها را به گروه‌های جداگانه یا خوشه‌ها تقسیم می‌کند. هدف اصلی خوشه‌بندی، یافتن ساختارهای مخفی در داده‌ها و تمایز بین دسته‌های مختلف است. روش‌های خوشه‌بندی برای فهمیدن الگوها، دسته‌بندی داده‌ها و بررسی رفتار آن‌ها در کنار هم، بسیار مفید هستند.

در این تمرین قرار است سه الگوریتم خوشه‌بندی اعم از K_Means ، K_Medians و DBSCAN مورد بررسی قرار گیرد.

۱- مراحل پیش‌پردازش

۱-۱- کد پایتون و خروجی‌ها

Import libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Reading data set-4.csv

In [2]:
```
df = pd.read_csv("data set-4.csv")
```

عکس ۱ ایمپورت کتابخانه‌ها و خواندن فایل csv

Out[3]:

	Weight	Cholesterol	Gender
0	102	111	1
1	115	135	1
2	115	136	1
3	140	167	0
4	130	158	1
...	...	...	...
542	172	207	1
543	129	157	1
544	107	115	1
545	117	147	1
546	148	176	1

547 rows × 3 columns

عکس ۲ نمایش جدولی و تعداد سطر و ستون دیتافریم

```
In [5]: df.dtypes

Out[5]: Weight         int64
        Cholesterol    int64
        Gender         int64
        dtype: object
```

عکس ۳ نوع‌داده‌ای ستون‌ها

```
In [6]: df.describe()
```

Out[6]:

	Weight	Cholesterol	Gender
count	547.000000	547.000000	547.000000
mean	143.572212	170.433272	0.513711
std	30.837275	39.147189	0.500269
min	95.000000	102.000000	0.000000
25%	116.000000	136.000000	0.000000
50%	140.000000	169.000000	1.000000
75%	171.000000	208.000000	1.000000
max	203.000000	235.000000	1.000000

عکس ۴ توصیفی از دیتافریم

Find the number of missing values for each columns

There are no missing values AND Inconsistent datas in this dataFrame

```
In [7]: df.isna().sum()
```

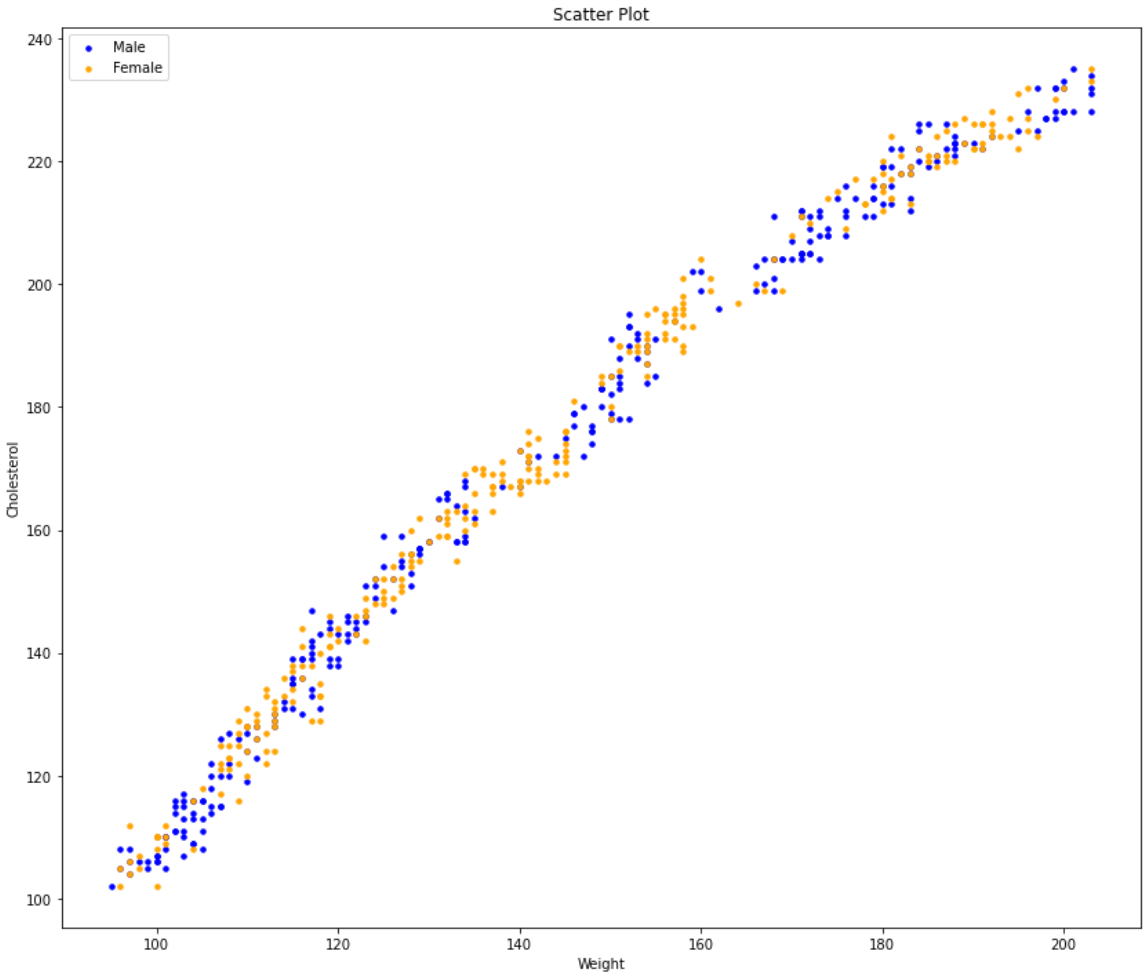
```
Out[7]: Weight         0
        Cholesterol    0
        Gender         0
        dtype: int64
```

عکس ۵ نمایش تعداد مقادیر از دسته رفته و بررسی مقادیر ناسازگار

در این دیتافریم مقادیر از دست رفته‌ای در ستون‌ها وجود ندارد و همچنین مقادیر ناسازگار نیز یافت نمی‌شود.

Create Scatter Plot to visualize data points relations

In [9]:
```
import matplotlib.pyplot as plt

# Filter the dataframe by gender
male_df = df[df['Gender'] == 1]
female_df = df[df['Gender'] == 0]

plt.figure(figsize=(14, 12))
# Create a scatter plot for male data points
plt.scatter(male_df['Weight'], male_df['Cholesterol'], c='blue', label='Male', s=12)

# Create a scatter plot for female data points
plt.scatter(female_df['Weight'], female_df['Cholesterol'], c='Orange', label='Female', s=12)

plt.xlabel('Weight')
plt.ylabel('Cholesterol')
plt.title('Scatter Plot')

plt.legend()
plt.show()
```



عکس ۶ رسم نمودار نقطه‌ای داده ها برای نمایش رابطه‌ی آن‌ها

۲- الگوریتم خوشه‌بندی K_Means

الگوریتم K_Means یکی از متداول‌ترین روش‌های خوشه‌بندی است که بر اساس مفهوم مراکز خوشه (centroid) عمل می‌کند. این الگوریتم به صورت تکراری و تفاضلی اجرا می‌شود تا به مراکز خوشه نهایی و درنهایت خوشه‌بندی داده‌ها برسد.

۲-۱- مراحل اصلی الگوریتم K_Means

۱. مشخص کردن تعداد خوشه‌ها(K) : ابتدا باید تعداد خوشه‌ها که می‌خواهیم داده‌ها را به آن‌ها تقسیم کنیم را مشخص کنیم.

۲. انتخاب مراکز خوشه اولیه: برای شروع، K نقطه (مرکز) را به صورت تصادفی به عنوان مراکز خوشه اولیه انتخاب می‌کنیم.

۳. نسبت دادن نقاط (داده‌ها) به خوشه‌ها: به ازای هر نقطه داده، نزدیک‌ترین مرکز خوشه را محاسبه کرده و آن نقطه را به آن خوشه اختصاص می‌دهیم.

۴. به‌روزرسانی مراکز خوشه: مراکز خوشه‌های جدید بر اساس میانگین نقاطی که به آن خوشه تعلق دارند، آپدیت می‌شوند.

۵. تکرار مراحل ۳ و ۴: مراحل ۳ و ۴ به صورت تکراری تا زمانی اجرا می‌شوند که مراکز خوشه‌ها تغییر نکنند یا معیاری که برای متوقف کردن الگوریتم درنظر گذفته شده است، ارضا شود. معمولاً مراکز خوشه‌ها در هر تکرار به مقدار اندکی تغییر می‌کنند.

۶. خروجی: خروجی نهایی الگوریتم K_Means شامل برچسب‌های خوشه برای هر نقطه داده است. همچنین، مراکز خوشه نهایی نیز قابل استفاده هستند و می‌توانند به عنوان نماینده‌های خوشه‌ها استفاده و نمایش داده شوند.

الگوریتم K_Means به دلیل استفاده از میانگین برای تعیین مراکز خوشه‌ها، حساسیت بالایی نسبت به نویز و داده‌های پرت دارد و نقاط و داده‌های تنها و پرت را به عنوان خوشه‌های جداگانه تشخیص نمی‌دهد.

sklearn KMeans Clustering

In [10]:
```python
from sklearn.cluster import KMeans

data = df[['Weight', 'Cholesterol', 'Gender']].values

num_clusters = 4

kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(data)

# Get the cluster assignments for each data point
assignments = kmeans.labels_

# Calculate the number of items in each cluster
cluster_counts = np.bincount(assignments)

# Calculate the mean of each cluster
cluster_centers = kmeans.cluster_centers_
```

عکس ۷ الگوریتم KMeans در کتابخانه sklearn

ابتدا مقادیر وزن و کلسترول و جنسیت را بدست می‌آوریم و تعداد خوشه‌ها (num_clusters) را برابر با ۴ قرار می‌دهیم.
حال الگوریتم KMeans را روی داده ها اعمال می‌کنیم و لیبل‌ها و مراکز خوشه‌ها را بدست می‌آوریم.

```
In [11]: # Print the number of items and mean of each cluster
total=0
for cluster in range(num_clusters):
    count = cluster_counts[cluster]
    total+=count
    mean = cluster_centers[cluster]
    print(f"Cluster {cluster+1}: {count} items, Mean: Weight = {mean[0]:.3f}, Cholesterol = {mean[1]:.3

print(f"\nTotal Number of Items: {total}")
```

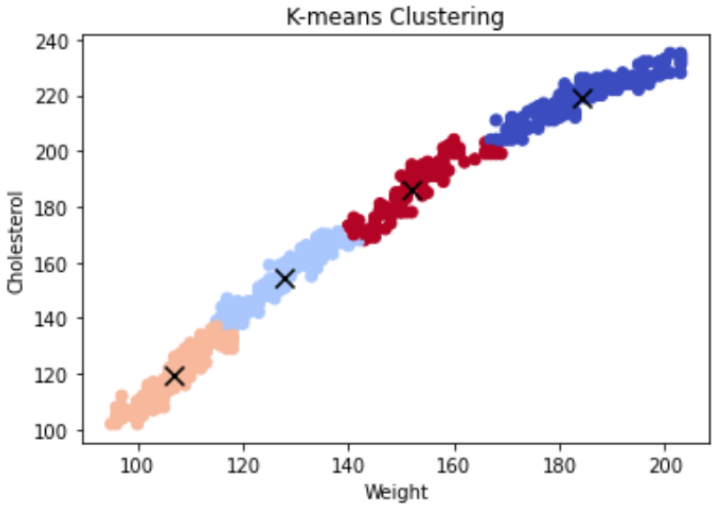
```
Cluster 1: 154 items, Mean: Weight = 184.318, Cholesterol = 218.916, Gender = 0.591
Cluster 2: 135 items, Mean: Weight = 127.726, Cholesterol = 154.385, Gender = 0.459
Cluster 3: 140 items, Mean: Weight = 106.850, Cholesterol = 119.536, Gender = 0.543
Cluster 4: 118 items, Mean: Weight = 152.093, Cholesterol = 185.907, Gender = 0.441

Total Number of Items: 547
```

عکس ۸ محاسبه تعداد اعضا به همراه میانگین ویژگی‌ها در هر خوشه (الگوریتم KMeans)

در این قسمت، تعداد اعضای هر خوشه به همراه میانگین سه ویژگی وزن، کلسترول و جنسیت در هر خوشه محاسبه می‌شود.

بیشترین میانگین این ویژگی‌ها را خوشه یک دارد؛ به طوریکه در این خوشه میانگین وزن برابر ۱۸۴.۳۱۸ ، میانگین کلسترول برابر ۲۱۸.۹۱۶ و میانگین جنسیت برابر ۰.۵۹۱ بدست آمده شده است. همچنین خوشه شماره یک دارای بیشترن اعضا با مقدار ۱۵۴ عضو است و خوشه شماره چهار دارای کمترین تعداد اعضا است.

```
In [12]: plt.scatter(data[:, 0], data[:, 1], c=assignments, cmap='coolwarm')#viridis
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='black', marker='x', s=100)
plt.xlabel('Weight')
plt.ylabel('Cholesterol')
plt.title('K-means Clustering')
plt.show()
```



عکس ۹ نمودار نقطه‌ای حاصل از خوشه‌بندی الگوریتم KMeans

از آنجایی که در این الگوریتم، تعداد خوشه‌ها (k) را برابر با عدد ۴ در نظر گرفتیم، داده‌ها به ۴ خوشه مطابق با عکس ۹ تقسیم شده‌اند. مراکز خوشه‌ها (Centroid) با علامت ضربدر و اعضای متعلق به هر خوشه با یک رنگ خاص مشخص شده‌اند.

۳- الگوریتم خوشه‌بندی K_Medians

الگوریتم خوشه‌بندی K-Medians یک الگوریتم خوشه‌بندی است که مشابه الگوریتم K-Means عمل می‌کند، با این تفاوت که به جای استفاده از میانگین برای آپدیت مراکز خوشه (centroid)، از مدیان خوشه استفاده می‌کند. این الگوریتم معمولاً در مواردی که داده‌ها توزیع‌شان پرتراکم و غیریکنواخت است، کارآمدتر از K-Means عمل می‌کند.

۳-۱- مراحل اصلی الگوریتم K_Medians

۱. مشخص کردن تعداد خوشه‌ها(K) : ابتدا باید تعداد خوشه‌ها که می‌خواهیم داده‌ها را به آن‌ها تقسیم کنیم را مشخص کنیم.

۲. انتخاب مراکز خوشه اولیه: برای شروع، K نقطه (مرکز) را به صورت تصادفی به عنوان مراکز خوشه اولیه انتخاب می‌کنیم.

۳. نسبت دادن نقاط (داده‌ها) به خوشه‌ها: به ازای هر نقطه داده، نزدیک‌ترین مرکز خوشه را محاسبه کرده و آن نقطه را به آن خوشه اختصاص می‌دهیم.

۴. به‌روزرسانی مراکز خوشه: مراکز خوشه‌های جدید بر اساس مدیان نقاطی که به آن خوشه تعلق دارند، آپدیت می‌شوند.

۵. تکرار مراحل ۳ و ۴: مراحل ۳ و ۴ به صورت تکراری تا زمانی اجرا می‌شوند که مراکز خوشه‌ها تغییر نکنند یا معیاری که برای متوقف کردن الگوریتم درنظر گذفته شده است، ارضا شود. معمولاً مراکز خوشه‌ها در هر تکرار به مقدار اندکی تغییر می‌کنند.

۶. خروجی: خروجی نهایی الگوریتم K_Medians شامل برچسب‌های خوشه برای هر نقطه داده است. همچنین، مراکز خوشه نهایی نیز قابل استفاده هستند و می‌توانند به عنوان نماینده‌های خوشه‌ها استفاده و نمایش داده شوند.

مزیت اصلی الگوریتم K_Medians نسبت به K_Means این است که مقاومت بیشتری در برابر داده‌های پرت و نویز دارد. بدین معنی که این الگوریتم، میانه‌ها را به جای مراکز خوشه‌ها استفاده می‌کند و به همین علت به ازای نقاط پرت یا نویز، تأثیر کمتری در محاسبات خوشه‌بندی دارد.

به طور کلی، الگوریتم K_Medians یک روش مفید برای خوشه‌بندی داده‌هاست، به خصوص در مواردی که داده‌ها توزیع‌شان پرتراکم و غیریکنواخت هستند. با این حال، باید به این نکته توجه کرد که محاسبه مدیان نسبت به محاسبه میانگین در K-Means ، محاسباتی بیشتری نیاز دارد و الگوریتم K-Medians معمولاً کندتر از K-Means اجرا می‌شود.

۳-۲- کد پایتون و خروجی‌ها

KMedians Clustering Implementation

In [13]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import pairwise_distances_argmin_min

def k_medians(X, n_clusters, max_iters=100):
    np.random.seed(40)
    centers = X[np.random.choice(X.shape[0], n_clusters, replace=False)]
    for _ in range(max_iters):
        # Assign each sample to the nearest cluster center based on euclidean distance
        labels = pairwise_distances_argmin_min(X, centers, metric='euclidean')[0]
        # Update cluster centers as medians of the samples in each cluster
        new_centers = np.array([np.median(X[labels == i], axis=0) for i in range(n_clusters)])

        # Check for convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return labels, centers
```

عکس ۱۰ پیاده‌سازی الگوریتم K_Medians

در این تابع، ابتدا به تعداد خوشه‌ها (n_clusters)، به صورت رندوم و از همان داده‌های مسئله، مراکز اولیه خوشه‌ها (centroid) مشخص می‌شوند. سپس با استفاده از فاصله اقلیدوسی، داده‌هایی که به هر یک از این مراکز نزدیک‌تر باشد را در خوشه متعلق به همان مرکز قرار می‌دهد. در نهایت با استفاده از مدیان اعضای هر خوشه، مراکز خوشه‌ها آپدیت می‌شود.

```
data = df[['Weight', 'Cholesterol', 'Gender']].values

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

# Set the number of clusters (k)
k = 4

# Apply k-medians clustering
labels, centers = k_medians(scaled_data, k)

# Print the number of items and mean of each cluster
num_clusters = len(centers)
cluster_counts = np.bincount(labels)
cluster_centers = scaler.inverse_transform(centers)
```

عکس ۱۱ اجرای الگوریتم K_Medians

ابتدا مقادیر وزن و کلسترول و جنسیت را بدست می‌آوریم و تعداد خوشه‌ها (num_clusters) را برابر با ۴ قرار می‌دهیم. حال الگوریتم KMedians را روی داده ها اعمال می‌کنیم و لیبل‌ها و مراکز خوشه‌ها را بدست می‌آوریم.

```
total = 0
for cluster in range(num_clusters):
    count = cluster_counts[cluster]
    total += count
    mean = cluster_centers[cluster]
    print(f"Cluster {cluster+1}: {count} items, Mean: Weight = {mean[0]:.3f}, Cholesterol = {mean[1]:.3

print(f"\nTotal Number of Items: {total}")
```

```
Cluster 1: 109 items, Mean: Weight = 179.000, Cholesterol = 214.000, Gender = 1.000
Cluster 2: 79 items, Mean: Weight = 105.000, Cholesterol = 116.000, Gender = 1.000
Cluster 3: 93 items, Mean: Weight = 132.000, Cholesterol = 159.000, Gender = 1.000
Cluster 4: 266 items, Mean: Weight = 140.000, Cholesterol = 169.000, Gender = 0.000

Total Number of Items: 547
```

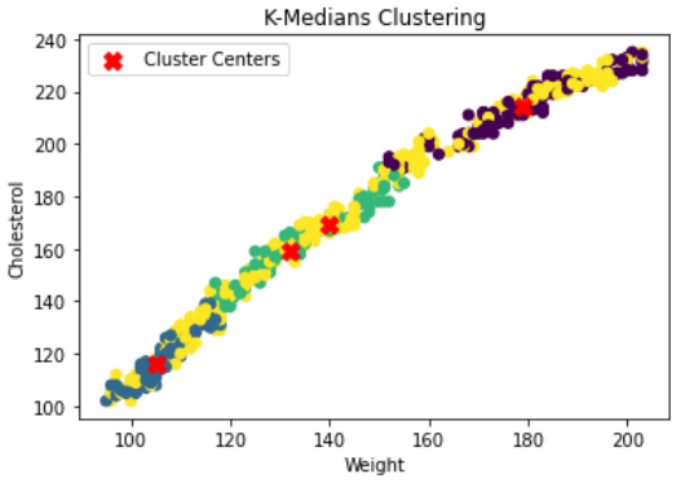
عکس ۱۲ محاسبه تعداد اعضا به همراه میانگین ویژگی‌ها در هر خوشه (الگوریتم KMedians)

در این قسمت، تعداد اعضای هر خوشه به همراه میانگین سه ویژگی وزن، کلسترول و جنسیت در هر خوشه محاسبه می‌شود.

بیشترین میانگین این ویژگی‌ها را خوشه یک دارد؛ به طوریکه در این خوشه میانگین وزن برابر ۱۷۹.۰۰۰ ، میانگین کلسترول برابر ۲۱۴.۰۰۰ و میانگین جنسیت برابر ۱ بدست آمده شده است.

خوشه شماره چهار دارای بیشترین اعضا با مقدار ۲۶۶ عضو است و خوشه شماره دو دارای کمترین تعداد اعضا است.

```
In [14]: # Extract the original unscaled data for plotting
unscaled_data = scaler.inverse_transform(scaled_data)
# Extract the coordinates of the cluster centers
unscaled_centers = scaler.inverse_transform(centers)
plt.scatter(unscaled_data[:, 0], unscaled_data[:, 1], c=labels)
plt.scatter(unscaled_centers[:, 0], unscaled_centers[:, 1], c='red', marker='X', s=100,
            label='Cluster Centers')
plt.xlabel('Weight')
plt.ylabel('Cholesterol')
plt.title('K-Medians Clustering')
plt.legend()
plt.show()
```



عکس ۱۳ نمودار نقطه‌ای حاصل از خوشه‌بندی الگوریتم KMedians

از آنجایی که در این الگوریتم، تعداد خوشه‌ها (k) را برابر با عدد ۴ در نظر گرفتیم، داده‌ها به ۴ خوشه مطابق با عکس ۱۳ تقسیم شده‌اند. مراکز خوشه‌ها (Centroid) با علامت ضربدر و اعضای متعلق به هر خوشه با یک رنگ خاص مشخص شده‌اند.

٤- الگوریتم خوشه‌بندی DBSCAN

الگوریتم DBSCAN (Density-Based Spatial Clustering of Applications with Noise) یک الگوریتم خوشه‌بندی است که بر اساس چگالی نقاط در فضای مختصاتی عمل می‌کند. این الگوریتم به خوشه‌بندی داده‌ها بر اساس توزیع نقاط در نزدیکی یکدیگر می‌پردازد و به صورت خودکار خوشه‌های چگال را تشخیص می‌دهد.

۴-۱- مراحل اصلی الگوریتم DBSCAN

۱. تعیین پارامترها: ابتدا باید دو پارامتر اصلی را تعیین کنیم:

Epsilon (ε) : این پارامتر برای تعیین شعاع همسایگی استفاده می‌شود. همسایگی یک نقطه شامل نقاطی است که فاصله آنها از نقطه مورد نظر کمتر یا مساوی با ε است.

MinPts یا min_samples : این پارامتر حداقل تعداد نقاطی است که باید در شعاع همسایگی یک نقطه قرار گیرند تا آن نقطه به عنوان یک نقطه هسته یا مرکز (centroid) شناخته شود.

۲. انتخاب نقطه اولیه: یک نقطه دلخواه را انتخاب کرده و آن‌را به عنوان نقطه جاری مشخص می‌کنیم.

۳. پیدا کردن همسایگان: تمام نقاطی که در شعاع همسایگی ε قرار می‌گیرند، به عنوان همسایگان نقطه جاری در نظر گرفته می‌شوند.

۴. بررسی تعداد همسایگان: اگر تعداد همسایگان نقطه جاری بیشتر از MinPts (min_samples) باشد، نقطه جاری به عنوان یک نقطه هسته یا مرکز شناخته می‌شود. در غیر این صورت، نقطه جاری به عنوان نقطه تکرار شونده مشخص می‌شود.

نکته : نقطه تکرار شونده (redundant point) در الگوریتم DBSCAN نقاطی هستند که تعداد همسایگان آنها کمتر از حداقل تعداد نقاط مورد نیاز برای تشکیل یک خوشه (MinPts) است. به عبارت دیگر، نقاطی که تعداد همسایگانشان از MinPts کمتر باشد، به عنوان نقاط تکرار شونده شناخته می‌شوند. نقاط تکرار شونده به دلیل تعداد همسایگان کمتر از MinPts، نمی‌توانند به عنوان نقاط هسته یا مرکز در خوشه‌بندی در نظر گرفته شوند و خوشه‌هایی را به وجود بیاورند.

۵. ادامه جستجو: بررسی تمام همسایگان هر نقطه هسته و تشکیل خوشه‌ها؛ اگر یک همسایه نقطه هسته نیز یک نقطه هسته باشد، آن دو نقطه به یک خوشه تعلق می‌گیرند.

۶. تشکیل خوشه‌های جدید: نقاطی که همسایه نقطه هسته نبوده و در شعاع همسایگی یک نقطه هسته قرار دارند، به عنوان نقاط مرزی تعیین می‌شوند و ممکن است به خوشه‌های موجود پیوسته و یا خوشه‌های جدید ایجاد کنند.

۷. تکرار: این مراحل برای تمام نقاط هسته و نقاط مرزی ادامه می‌یابد تا همه نقاط داده پردازش شوند و تمام خوشه‌ها، شناسایی شوند.

۸. نقاط اشتباه و نویز: نقاطی که همسایه نقطه هسته نیستند و در شعاع همسایگی هیچ نقطه‌ی هسته‌ای قرار ندارند، به عنوان نقاط نویز شناسایی می‌شوند و به هیچ خوشه‌ای تعلق نمی‌گیرند.

نتیجه نهایی الگوریتم DBSCAN شامل چند خوشه است که هر خوشه شامل یک یا چند نقطه هسته و نقاط مرزی است.

همچنین الگوریتم DBSCAN قادر است به صورت خودکار نقاط نویز را تشخیص دهد و آنها را از هر خوشه جدا کند.

الگوریتم DBSCAN دارای ویژگی‌هایی مانند مقاومت در برابر نویز، تشخیص خوشه‌های با اشکال متفاوت و عدم نیاز به تعیین تعداد خوشه‌ها از قبل است. اما این الگوریتم به پارامترهای ε و MinPts حساس است و نتایج آن ممکن است به شدت تحت تأثیر این پارامترها قرار گیرند. بنابراین، تنظیم صحیح این پارامترها برای داده‌های مختلف بسیار مهم است.

۴-۲- کد پایتون و خروجی‌ها

Task 5 : DBSCAN

In [15]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

data = df[['Weight', 'Cholesterol', 'Gender']].values

# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data)

dbscan = DBSCAN(eps=0.08, min_samples=10)

dbscan.fit(scaled_features)

# Retrieve the cluster labels and unique labels
labels = dbscan.labels_
unique_labels = set(labels) - {-1}
```

عکس ۱۴ اجرای الگوریتم DBSCAN

مرحله ۱ : ابتدا مقادیر وزن و کلسترول و جنسیت را بدست می‌آوریم.

مرحله ۲ (استاندارد سازی ویژگی‌ها (Standardize the features)) : در این مرحله، از کلاس StandardScaler کتابخانه‌ی scikit-learn برای استانداردسازی ویژگی‌ها استفاده می‌شود. با استفاده از این کلاس، مقادیر ویژگی‌ها تبدیل و استانداردسازی می‌شوند؛ به طوری که میانگین صفر و واریانس یک داشته باشند. نتیجه این است که ویژگی‌ها به یک مقیاس مشابه تبدیل می‌شوند. این کار باعث می‌شود که مقیاس واحدی برای تمام ویژگی‌ها برقرار شود. به عبارت دیگر، اگر مقداری برای یک ویژگی بزرگتر از مقداری دیگر باشد، در فرآیند استانداردسازی، این تفاوت مقیاس حذف می‌شود و همه ویژگی‌ها به یک مقیاس مشابه تبدیل می‌شوند. در نتیجه، الگوریتم DBSCAN بر روی داده‌های استانداردسازی شده اجرا می‌شود. به دلیل استانداردسازی ویژگی‌ها، هیچ ویژگی به طور مستقیم از بقیه ویژگی‌ها برتری یا تأثیر بیشتری ندارد. این مورد باعث می‌شود الگوریتم DBSCAN به طور متوسط بر روی همه‌ی ویژگی‌ها تأثیر بگذارد و به نحوی خوشه‌بندی را انجام دهد.

مرحله ۳ (ساخت نمونه‌ای از کلاس DBSCAN) : در این مرحله، یک نمونه از کلاس DBSCAN ایجاد می‌شود. در این نمونه، مقادیر پارامترهای eps و min_samples تنظیم می‌شوند. eps نشان دهنده شعاع همسایگی است که برای تعیین همسایگان هر نقطه استفاده می‌شود و min_samples نشان دهنده حداقل تعداد نقاطی است که باید در شعاع همسایگی یک نقطه وجود داشته باشند تا آن نقطه به عنوان یک نقطه هسته یا مرکز شناخته شود.

مرحله ۴ (فیت کردن مدل) : با فراخوانی متد fit روی نمونه DBSCAN ، الگوریتم شروع به خوشه‌بندی داده‌های استانداردسازی شده می‌کند.

مرحله ۵ : در این مرحله پس از اجرای الگوریتم DBSCAN ، برچسب‌های خوشه‌ها برای هر نقطه در labels ذخیره می‌شوند. همچنین با استفاده از مجموعه set و عملکرد تفاضل مجموعه‌ها {1-} - ، برچسب‌های خوشه‌ها را در unique_labels ذخیره می‌کنیم. نکته مهم این است که برچسب ۱- به نقاط نویز تخصیص داده می‌شود و این کار برای نقاطی که در هیچ خوشه‌ای قرار نمی‌گیرند، انجام می‌شود.

```
In [16]: total = 0
for cluster in unique_labels:
    count = (labels == cluster).sum()
    total += count
    cluster_data = data[labels == cluster]
    mean = np.mean(cluster_data, axis=0)

    print(f"Cluster {cluster}: {count} items, Mean: Weight = {mean[0]:.3f}, Cholesterol = {mean[1]:.3f}, Gender = {mean[2]:.3f}")

print(f"\nTotal Number of Items: {total}")
```

```
Cluster 0: 31 items, Mean: Weight = 103.548, Cholesterol = 113.097, Gender = 1.000
Cluster 1: 14 items, Mean: Weight = 191.286, Cholesterol = 224.357, Gender = 0.000
Cluster 2: 21 items, Mean: Weight = 156.619, Cholesterol = 193.952, Gender = 0.000
Cluster 3: 28 items, Mean: Weight = 139.250, Cholesterol = 168.893, Gender = 0.000
Cluster 4: 10 items, Mean: Weight = 108.100, Cholesterol = 123.100, Gender = 0.000
Cluster 5: 19 items, Mean: Weight = 170.895, Cholesterol = 205.474, Gender = 1.000
Cluster 6: 15 items, Mean: Weight = 111.200, Cholesterol = 128.333, Gender = 0.000
Cluster 7: 13 items, Mean: Weight = 98.846, Cholesterol = 106.077, Gender = 1.000
Cluster 8: 10 items, Mean: Weight = 185.900, Cholesterol = 220.800, Gender = 0.000
Cluster 9: 10 items, Mean: Weight = 132.300, Cholesterol = 161.000, Gender = 0.000
Cluster 10: 11 items, Mean: Weight = 125.182, Cholesterol = 150.000, Gender = 0.000
Cluster 11: 10 items, Mean: Weight = 152.400, Cholesterol = 190.500, Gender = 1.000
Cluster 12: 12 items, Mean: Weight = 121.167, Cholesterol = 144.250, Gender = 1.000

Total Number of Items: 204
```

عکس ۱۵ محاسبه تعداد اعضا به همراه میانگین ویژگی‌ها در هر خوشه (الگوریتم DBSCAN)

در این قسمت، تعداد اعضای هر خوشه به همراه میانگین سه ویژگی وزن، کلسترول و جنسیت در هر خوشه محاسبه می‌شود.

بیشترین میانگین این ویژگی‌ها را خوشه یک دارد؛ به طوریکه در این خوشه میانگین وزن برابر ۱۹۱.۲۸۶ ، میانگین کلسترول برابر ۲۲۴.۳۵۷ و میانگین جنسیت برابر ۰ بدست آمده شده است.

خوشه شماره سه دارای بیشترن اعضا با مقدار ۲۸ عضو است و خوشه شماره هشت و یازده دارای کمترین تعداد اعضا یعنی ۱۰ عضو هستند.

در این الگوریتم، مقدار eps برابر ۰.۰۸ و min_samples برابر ۱۰ درنظر گرفته شده است.

از آنجایی که مقدار min_samples برابر با ۱۰ است، پس حداقل تعداد اعضای در هر خوشه باید برابر با ۱۰ عضو باشد و به همین دلیل است که خوشه‌های ۸ و ۱۱ شامل ۱۰ عضو هستند و این مقدار کمترین مقدار ممکن برای تعداد اعضای متعلق به هر خوشه است.

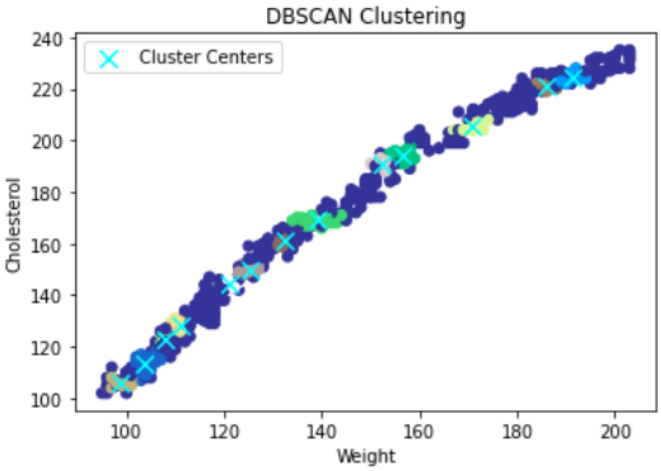
با افزایش مقدار min_samples در صورت ثابت بودن مقدار eps ، تعداد خوشه‌ها کاهش و تعداد داده‌های پرت افزایش می‌یابد؛ همچنین با کاهش مقدار min_samples تعداد خوشه‌ها افزایش و تعداد داده‌های پرت کاهش می‌یابد.

با افزایش مقدار eps در صورت ثابت بودن مقدار min_samples ، تعداد خوشه‌ها و تعداد داده‌های پرت کاهش می‌یابد؛ همچنین با کاهش مقدار eps تعداد خوشه‌ها و تعداد داده‌های پرت افزایش می‌یابد.

توجه : اما توضیحات فوق ممکن است بسته به یک مقدار خاص از eps و یا min_samples و تغییر آن‌ها، همچنین با توجه به داده‌های مسئله، درست نباشد و نتوان چنین نتیجه‌ای گرفت.

تعداد کل داده‌ها در دیتافریم این تمرین برابر ۵۴۷ است (۵۴۷ سطر دارد) اما اگر توجه کرده باشید، در الگوریتم DBSCAN و عکس ۱۵ ، تعداد کل داده‌ها که به خوشه‌ها تعلق دارند، برابر ۲۰۴ است؛ این به این معنا است که با تعیین min_samples = 10 و eps = 0.08 ، ۲۰۴ – ۵۴۷ یعنی ۳۴۳ داده پرت شناسایی شده است که در هیچ خوشه‌ای قرار نگرفته‌اند؛ همانطور که پیشتر توضیح داده شد، الگوریتم DBSCAN قادر است به صورت خودکار داده‌های نویز را تشخیص دهد و آنها را از هر خوشه جدا کند.

```
# Compute cluster centers
centers = []
for cluster in unique_labels:
    cluster_data = data[labels == cluster]
    cluster_center = np.mean(cluster_data, axis=0)
    centers.append(cluster_center)
centers = np.array(centers)
# Plot the clusters and centers
plt.scatter(data[:, 0], data[:, 1], c=labels , cmap='terrain')
plt.scatter(centers[:, 0], centers[:, 1], marker='x', c='cyan', s=100, label='Cluster Centers')
plt.xlabel('Weight')
plt.ylabel('Cholesterol')
plt.title('DBSCAN Clustering')
plt.legend()
plt.show()
```



عکس ۱۶ نمودار نقطه‌ای حاصل از خوشه‌بندی الگوریتم DBSCAN

از آنجایی که در این الگوریتم، min_samples = 10 و eps = 0.08 در نظر گرفتیم، داده‌ها به ۱۳ خوشه مطابق با عکس ۱۶ تقسیم شده‌اند. مراکز خوشه‌ها (Centroid) با علامت ضربدر و اعضای متعلق به هر خوشه با یک رنگ خاص مشخص شده‌اند که در تصویر ۱۶ ، ۱۳ رنگ متفاوت وجود دارد.