DBSCAN Algorithm, 1399/11/06

تمرین دوم درس شناسایی الگو

نام : حسین سیم چی

۹۸۴۴۳۱۱۹

استاد : آقای دکتر احمدعلی آبین

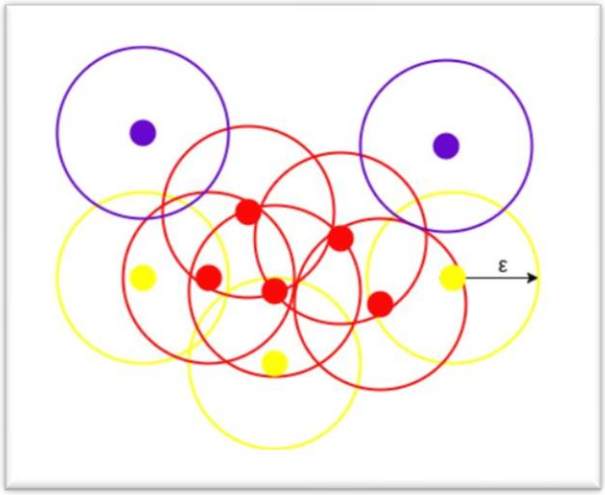
مقدمه

در تمرین دوم قصد داریم نحوه ی پیاده سازی الگوریتم DBSCAN را شرح دهیم، سپس با بیان نقاط قوت و ضعف این الگوریتم، آن را به گونه ای توسعه دهیم تا موارد ضعف را در بسیاری از دیتاست ها و اشکال موجود برطرف نماید.

شرح کار الگوریتم DBSCAN

در ابتدا فرض کنید که داده های ما در فضای دو بعدی (تنها دو بعد) مانند شکل زیر در فضا پخش شده اند:

حال برای انجام الگوریتم DBSCAN باید در اطراف هر داده شعاع مشخصی را درنظر بگیریم. با شروع از داده ی اول و درنظر گرفتن شعاع برای داده ی نخست، برچسب خاصی را نیز برای آن درنظر می گیریم. سپس برای داده ی دوم نیز دایره ای با شعاع مشخصی را لحاظ میکنیم، اگر فاصله ی داده ی دوم تا داده ی اول از شعاع درنظر گرفته شده کمتر باشد الگوریتم DBSCAN برچسب داده ی اول را به داده ی دوم اختصاص می دهد درغیر این صورت داده ی دوم فاصله ی درنظر گرفته شده را رعایت نکرده (از شعاع بیشتر شده) و در نتیجه ی آن برچسب جدیدی را دریافت می کند. که در شکل زیر، نمای بسیار ساده ای برای داده های فوق جهت درک بهتر رسم شده است



همانطور که از شکل فوق نیز مشخص است، بر اساس شعاع و توضیحات داده شده هر داده برچسب خاصی را به خود اختصاص می دهد.

با توجه به توضیحات داده شده الگوریتم DBSCAN دارای دو پارامتر اساسی و اصلی است

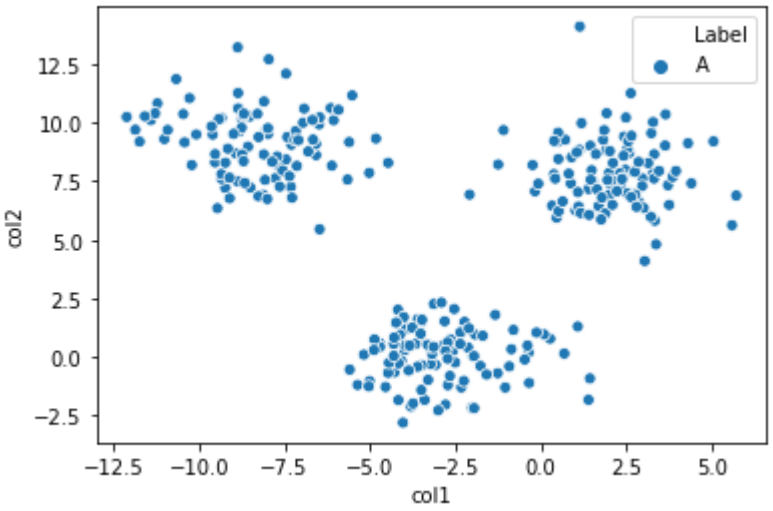
۱. Min points : باید برای هر کلاستر تعداد نقاط یا داده ی مشخصی را درنظر بگیریم، به این صورت که به عنوان مثال برای هر کلاستر، فاصله ی تعداد داده های مشخصی را تا مرکز کلاستر لحاظ کرده و شرط فاصله را بررسی می کنیم. بدیهی است که تعداد نقاطی که فاصله ی آن ها را محاسبه کرده یا به عبارتی درون هر کلاستر قرار می گیرند رابطه ی مستقیم با شعاع درنظر گرفته شده دارد که هرچه شعاع بیشتر باشد درنتیجه تعداد داده های بیشتری درون هر کلاستر قرار می گیرند و تعداد کلاسترهای کمتری را نیز خواهیم داشت.
۲. Eps : مقدار اپسیلون نشان دهنده ی شعاع درنظر گرفته شده برای هر داده می باشد. درنتیجه هرچه شعاع بیشتر باشد، تعداد کلاستر کمتری نیز خواهیم داشت.

مزایای الگوریتم DBSCAN

۱. در کمترین زمان ممکن می تواند خوشه بندی را انجام دهد

۲. سادگی

۳. مهم ترین مزیت این الگوریتم زمانی است که داده ها به صورت یکنواخت در فضا پخش نشده اند و چگالی داده ها در فضای ابعاد متفاوت است. به عنوان مثال شکل زیر را درنظر بگیرید که داده ها به صورت متمرکز در سه نقطه از فضا پخش شده اند



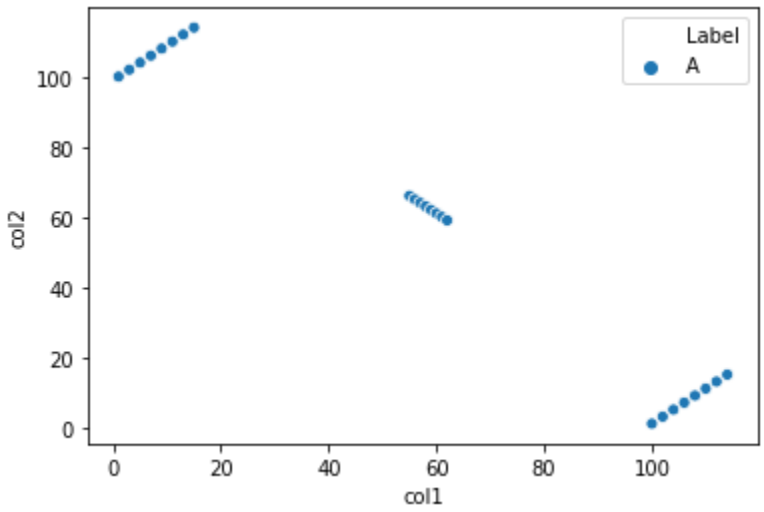
اگر از الگوریتم های دیگر مطرح شده در زمینه ی خوشه بندی مانند K Means استفاده کنیم خواهیم دید که خوشه بندی درستی صورت نمی گیرد چون به عنوان مثال در الگوریتم K Means میانگین داده ها را درنظر می گیریم پس در نتیجه نقاطی که نویز هستند تاثیر بسیار زیادی در مرکز خوشه و به تبع آن خوشه بندی خواهند گذاشت. در صورتی که در این الگوریتم این اتفاق نمی افتد و رفتار بسیار درستی با داده های نویز صورت می گیرد.

۴. داده های نویز تاثیری بر روی خوشه بندی نخواهند داشت (به دلیل اینکه قاعدتا در فاصله های دورتری قرار میگیرند و به تبع آن مقدار فاصله ی زیادی با داده های دیگر دارند و درون کلاستر جداگانه ای قرار خواهند گرفت).

کد نوشته شده برای اجرای الگوریتم DBSCAN

کتابخانه ی Sklearn در زبان برنامه نویسی پایتون، الگوریتم DBSCAN را به صورت آماده دارد و کافی است دو پارامتر ذکر شده را برای آن مشخص کنیم تا خوشه بندی را انجام دهد. در این تمرین از **نتایج (نه کد)** این کتابخانه برای خوشه بندی استفاده می کنیم و درنهایت با نتایج حاصل از کد نوشته شده و توسعه یافته ی آن مقایسه خواهیم کرد

با نگاه بسیار ساده در ابتدا سعی می کنیم الگوریتم موردنظر را حل و پیاده سازی کنیم. درنتیجه نیاز داریم تا داده هایی را تعریف کنیم که دارای دو بعد (دو ویژگی) باشند و در ادامه با رسم این داده ها در فضای ویژگی مانند شکل زیر، چگالی پخش آن ها از یکدیگر متفاوت باشد. یا به عبارتی دیگر در قسمت های مختلف از فضای ویژگی فشردگی داده ها با یکدیگر متفاوت باشد. پس بدون درنظر گرفتن دیتاست های مطرح، دیتاست ساده ای را تعریف می کنیم که به صورت زیر می باشد:



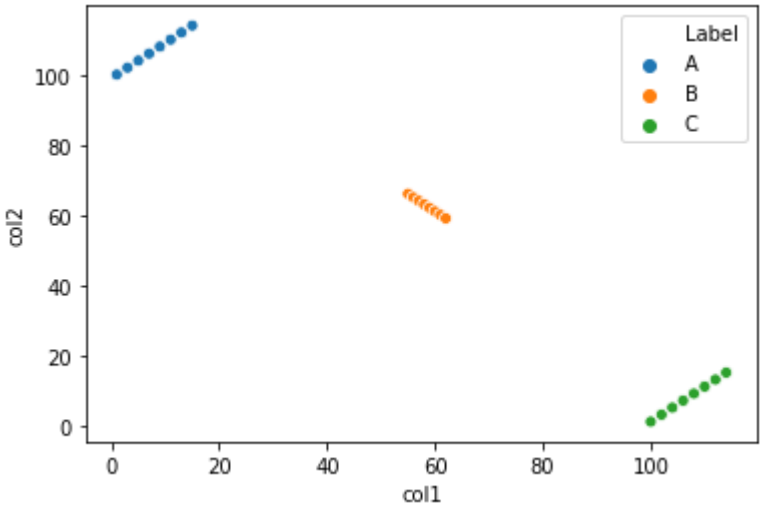
***** برای تعریف دیتاست استفاده شده، اعداد دلخواه داده شده اند و از کتابخانه ی Pandas برای تعریف دیتاست و انجام عملیات بر روی آن استفاده شده است.**

حال نیاز داریم به ازای تک تک داده های موجود در دیتاست به ترتیب شعاع خاصی را درنظر بگیریم و مراحل توضیح داده شده در ابتدای گزارش را بر روی آن انجام دهیم *که کد ساده ای برای همین منظور نوشته شده* است که زیر آمده است:

```
def  radius(x,y):
        a = x[0]
        b = x[1]
        c = y[0]
        d = y[1]
        distance = np.sqrt((a-c)**2 + (b-d)**2)    >>>  محاسبه ی فرمول دایره
        return distance


for i in range(len(points)):
   if (i+1) != len(points):
        x = points[i]
        y = points[i+1]
        distance = radius(x,y)  >>> محاسبه ی فاصله ی هر داده تا داده ی قبلی
        if distance < 100 :  >>>   محاسبه ی شرط شعاع برای هر داده
                label.append(label[i]) >>> اگر برقرار باشه، برچسب داده ی قبل خود را دریافت می کند
         else :
                label.append(pre_l[0]) >>> درغیر این صورت برچسب جدیدی را دریافت می کند.
                pre_l.remove(pre_l[0])
```

اگر مقدار شعاع درستی را انتخاب کنیم، خروجی الگوریتم فوق مانند شکل زیر خواهد بود:



تا به اینجای کار متوجه شده اید که یکی از چالش های موجود در این الگوریتم انتخاب صحیح مقدار پارامترهای مطرح شده است.

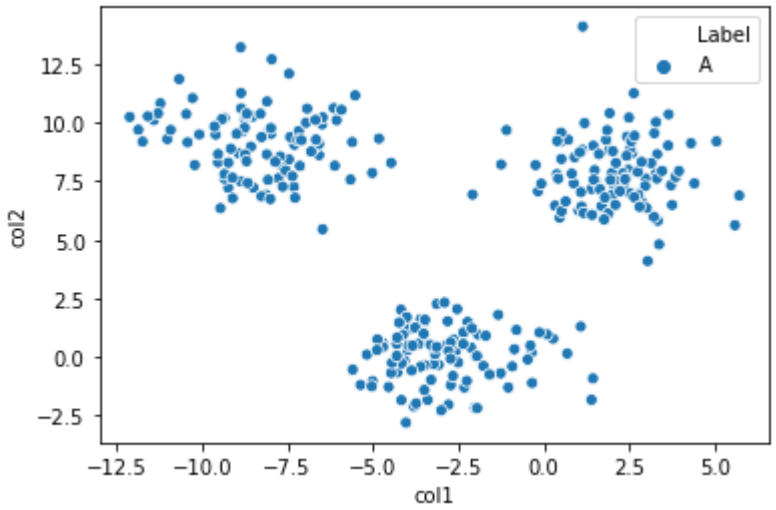
تا به اینجا سعی بر آن شد که به صورت خیلی ساده الگوریتم نام برده شده را اجرا کنیم. در ادامه می خواهیم بر روی دیتاست های پیچیده، **الگوریتم DBSCAN (ساده و توسعه نیافته)** را اجرا کنیم

***** کدنوشته شده برای اجرا این الگوریتم با نام Simchi_DBSCAN_First_View در پوشه تمرین قرار داده شده است**

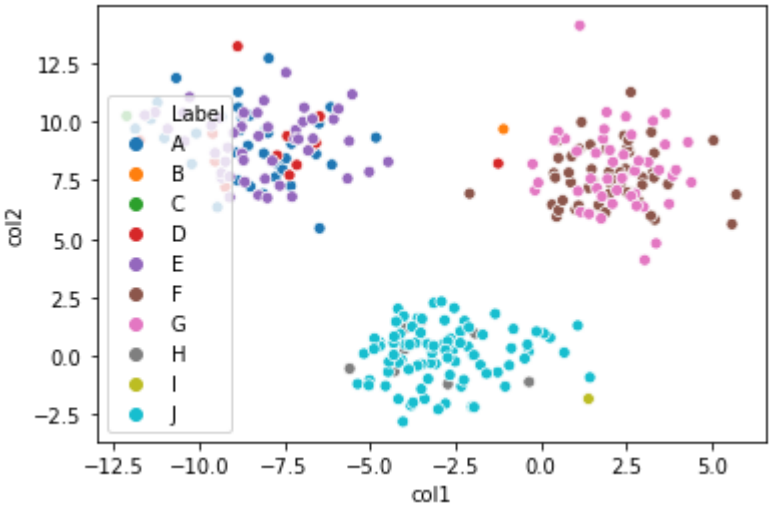
اجرای الگوریتم DBSCAN بر روی دیتاست های پیچیده

با استفاده از دیتاست های موجود در اینترنت و دیتاست های تعریف شده، در ابتدا الگوریتم نوشته شده بر روی آن ها اجرا می کنیم و سپس با طرح مشکلات الگوریتم، سعی می کنیم طوری آن را توسعه دهیم که مشکلات تا حد ممکن از بین برود.

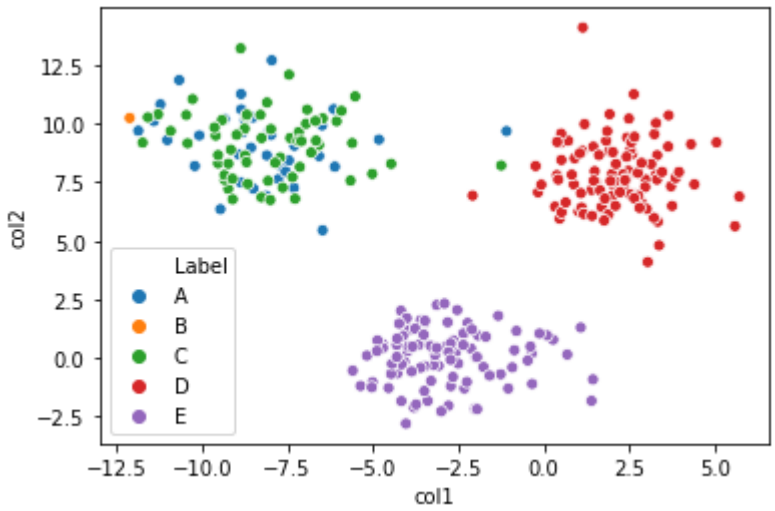
اگر کد نوشته شده و بررسی شده در قسمت قبل را بر روی دیتاست اول اجرا کنیم، در صورتی که شعاع را بزرگ درنظر بگیریم همگی داده ها در یک خوشه قرار می گیرند (شکل زیر)



حال اگر شعاع را از حدی کوچکتر درنظر بگیریم ممکن است به ازای هر داده یک کلاس داشته باشیم (شکل زیر)



ولی واقعا شعاع ایده آل چیست ؟ برای پاسخ با این سوال می توان به صورت سعی و خطا عمل کرد، یعنی خطا به ازای هر مقدار دلخواه شعاع را محاسبه کنیم و در نهایت به ازای چندین بار تکرار، شعاعی را انتخاب کنیم که باعث مینیمم شدن خطا شود. در نتیجه اگر شعاع ایده آل را بدست آوریم شکل زیر بدست خواهد آمد:

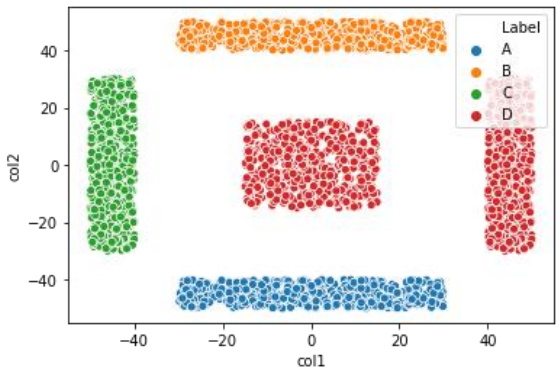


همانطور که از شکل فوق قابل برداشت است، ایراد الگوریتم DBSCAN کاملا مشهود است.

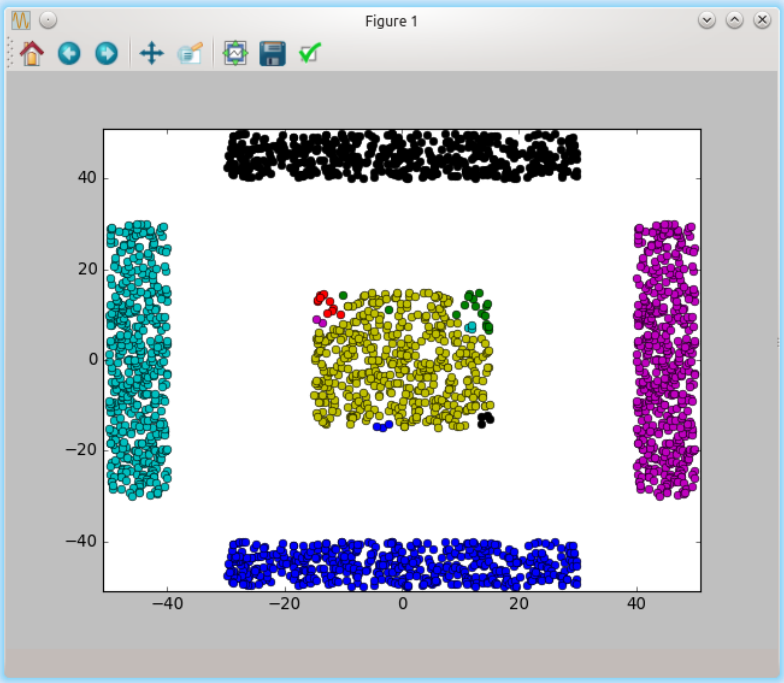
ایراد روش DBSCAN

داده ها با چگالی های متفاوت در فضای ویژگی پخش شده اند و **اگر میزان چگالی داده ها از حدی بیشتر باشد الگوریتم به خوبی عمل میکند در غیر این صورت ممکن است مانند شکل فوق برای یک کلاستر چندین نوع برچسب تشخیص داده شوند. به عبارتی دیگر اگر داده ها فاصله ای بسیار نزدیک باهم در هر کلاستر داشته باشند همگی یک برچسب میگیرند (رعایت شعاع) ولی اگر فاصله ی داده ها در یک کلاستر با هم متفاوت باشد (مانند شکل فوق)، ممکن است به اشتباهی چندین برچسب به یک کلاستر اختصاص داده شود.**

شکلی دیگر از نتایج الگوریتم DBSCAN که بر روی دیتاست دیگر انجام شده است در زیر قابل مشاهده است



همانطور که از شکل فوق قابل مشاهده است باید ۵ کلاستر داشته باشیم درحالی که خروجی ۴ کلاستر را نشان می دهد. علت این مشـکل به ایراد دوم روش DBSCAN برمی گردد. ایراد دوم این روش زمانی اتفاق می افتد که داده ها در فضای ویژگی حالت تقارن داشته باشند مانند چیزی که در شکل فوق می بینیم. در این حالت به دلیل اسـتفاده از معیار فاصـله ی دایره، منفی یا مثبت بودن داده ها با یکدیگر خنثی شـده و به اشـتباه در یک طرف از فضـای ویژگی برچسـب یکسـانی خواهیم داشـت. در شـکل زیر خروجی الگوریتم موجود در کتابخانه ی Sklearn را بر روی دیتاست فوق را مشاهده می کنید.



همانطور که از خروجی فوق نیز مشخص است، خروجی کتابخانه ی Sklearn نیز نتوانسته به خوبی خوشه بندی را انجام دهد ولی از کد نوشته شده ی ما بهتر عمل کرده است. دلیل این اتفاق چیزی نیست به جز دقت برنامه نوسی!!!.

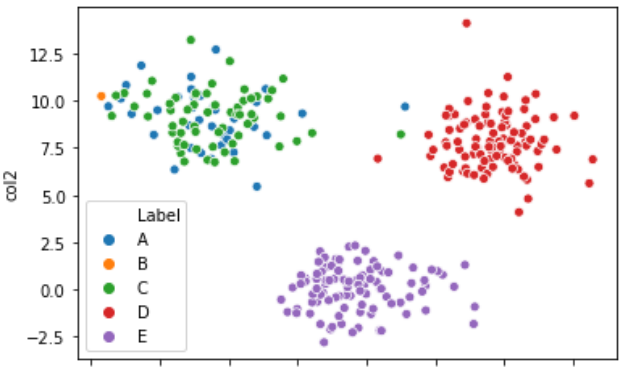
ولی با اینحال کدی که ما نوشته ایم و کدی که از کتابخانه میباشد هردو دارای نواقصی هستند که سعی می کنیم در ادامه آن را برطرف کنیم.

اجرای الگوریتم DBSCAN توسعه یافته بر روی دیتاست های پیچیده

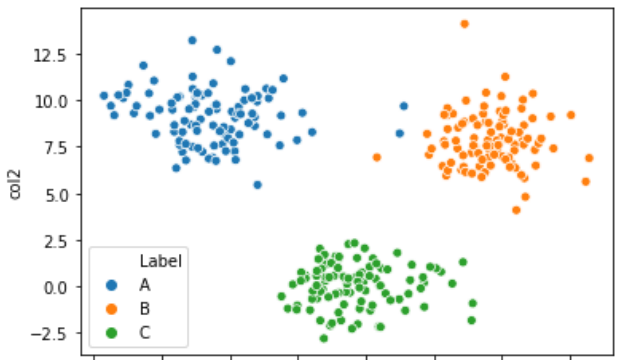
ایده ی اصلی باتوجه به مشکلاتی که بیان کرده ایم بسیار ساده است. مشکل اصلی این الگوریتم زمانی بوجود می آمد که در یک کلاستر فاصله ی داده ها با یکدیگر یکسان و یا از حدی مشخص کوچکتر نبود. در این صورت همانطور که در قسمت قبل با یکدیگر صحبت کردیم ممکن است در یک کلاستر چندین برچسب برای داده ها داشته باشیم.

ایده ی اصلی برای توسعه ی این الگوریتم، متغیر دانستن شعاع در فضای ویژگی است

متغیر بودن شعاع بدین معنا است که در قسمت هایی از فضای ویژگی که داده ها فاصله های بیشتری با یکدیگر دارند شعاع بزرگ تر و در قسمت های دیگر شعاع کوچکتری را درنظر بگیریم. برای انجام این الگوریتم کافی است درابتدا با استفاده از Preprocessing نمایشی از داده ها داشته باشیم و در ادامه با مشاهده ی داده ها ، فضای ویژگی ها را به تعدادی قسمت تقسیم کنیم که در هر قسمت بر اساس نحوه ی نمایش داده ها شعاع خاصی را درنظر خواهیم گرفت. اگر ایده ی فوق را انجام دهیم شکل های بدست آمده در قسمت ها ی قبلی به صورت زیر تغییر پیدا می کنند.



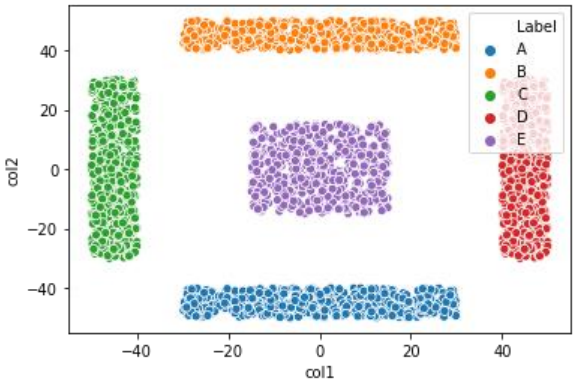
الگوریتم DBSCAN اولیه



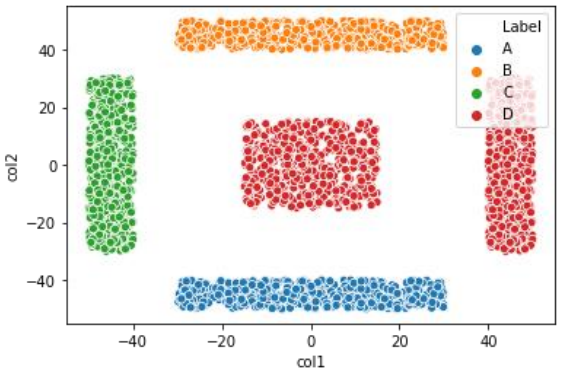
الگوریتم DBSCAN توسعه یافته

با مشاهده و مقایسه دو شکل فوق متوجه می شویم که توانسته ایم مشکل بیان شده برای الگوریتم را برطرف کرده و به دقت بالایی برای آن برسیم.

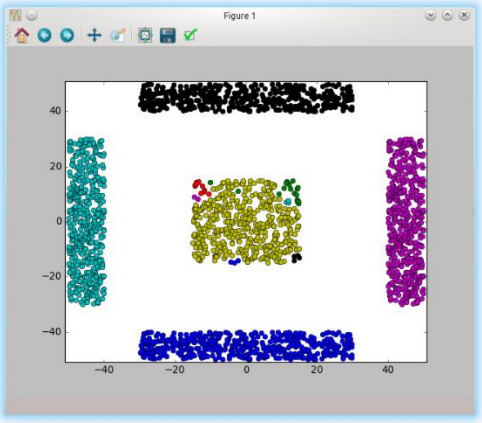
همچنین برای دیتاست دیگر نیز خواهیم داشت:



الگوریتم DBSCAN توسعه یافته



الگوریتم DBSCAN اولیه



الگورتیم DBSCAN مربوط به Sklearn

کاملا مشهود است که خروجی الگوریتم DBSCAN نوشته شده ی ما حتی از خروجی Sklearn هم بهتر کار میکند که به نوعی جالب توجه و قابل تامل است.

توجه: الگوریتم های توضـیح داده شـده بر روی ۲ دیتاسـت دیگر اجرا گرفته شـده اسـت که برای جلوگیری از افزایش حجم فایل گزارش از آوردن آن ها و توضیحات اضافه خودداری شده است.

تمامی موارد گفته شـده در فایل ارسـالی تمرین آورده شـده اسـت. کد نوشـته شـده، حاصـل تلاش های اینجانب بوده است و از هیچ گونه فایل کمکی استفاده نشده است.

باتشکر، حسین سیم چی

۶ بهمن ۱۳۹۹