به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

یادگیری ماشین

تکلیف پنجم

استاد درس: دکتر ناظرفرد

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

Contents

Q1)

a)

روش‌های خوشه بندی به دنبال یک بازنمایی از نحوه توزیع داده‌ها بدون در نظر گرفتن برچسب‌های آنها هستند، به عبارت دیگر، آنها داده‌ها را با در نظر گرفتن ویژگی‌های مختلف خوشه بندی می‌کنند. از طرف دیگر، اگر داده ای پرت باشد، به دلیل اینکه ویژگی‌های متفاوتی نسبت به سایرین دارد، نمی‌توان آن را به خوبی در یک از خوشه‌ها قرار داد، بنابراین داده‌هایی که در دسته خوشه‌های یافت شده با روش خوشه بندی ما قرار نمی‌گیرند با داده‌های دیگر تفاوت معناداری دارند را می‌توان به عنوان داده پرت تعریف نمود.

به بیان دیگر تشخیص داده پرت فرآیندی برای شناسایی نقاط داده ای است که به طور قابل توجهی با سایر نقاط در یک مجموعه داده متفاوت هستند. استراتژی‌های مختلفی برای آن وجود دارد، از جمله روش‌های مبتنی بر آمار، روش‌های مبتنی بر مجاورت و روش‌های مبتنی بر مدل. روش‌های مبتنی بر آمار، مانند Z-score یا روش Z-score اصلاح شده، از معیارهای آماری مانند میانگین و انحراف معیار برای شناسایی نقاط پرت استفاده می‌کنند. روش‌های مبتنی بر مجاورت، مانند روش k-نزدیک‌ترین همسایه، نقاط پرت را بر اساس فاصله هر نقطه داده از k نزدیک‌ترین همسایه‌اش شناسایی می‌کنند. روش‌های مبتنی بر مدل، مانند خوشه‌بندی مبتنی بر چگالی، از مدل‌های یادگیری ماشینی برای شناسایی نقاط پرت بر اساس انحراف از توزیع کلی داده‌ها استفاده می‌کنند. این استراتژی‌ها را می‌توان با هر روش خوشه‌بندی یا الگوریتمی برای تشخیص پرت استفاده کرد. این روش‌ها بر این باورند که اگر داده ای پرت باشد علاوه بر این که از بقیه‌ها به دور است از مراکز دسته‌ها نیز دور است و در نتیجه با معیارهای مرتبط با هر روش می‌توان آن را شناسایی نمود.

b)

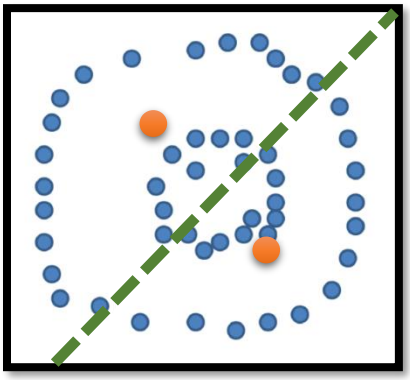
الگوریتم انتظار Expectation-Maximization (EM) یک روش آماری است که برای تخمین تابع به نحوی که بیشترین شباهت ممکن به توزیع داده‌ها را داشته باشد(maximum likelihood). این روش یک الگوریتم تکراری است که با حدس اولیه پارامترها شروع می‌شود و سپس تخمین‌ها را در هر تکرار به روز می‌کند و در هر تکرار مقادیر واریانس و میانگین آپدیت می‌شود.(دو گام دارد در گام اول (E-step) نقاط مربوط به هر دسته انتخاب می‌شوند در گام دوم(M-step) ویژگی ها آبدیت می‌شوند)

K-Meansشکل خاصی از الگوریتم EM است که از ترکیبی از توزیع‌های K Gaussian برای مدل‌سازی داده‌ها استفاده می‌کند. الگوریتم با انتخاب تصادفی K نقطه از داده‌ها به عنوان مرکز اولیه شروع می‌شود و سپس هر نقطه داده را به نزدیکترین مرکز می‌دهد. سپس میانگین هر خوشه بر اساس نقاط اختصاص داده شده مجددا محاسبه می‌شود و مرحله تخصیص تا زمان همگرایی تکرار می‌شود.

درK-Means ، کوواریانس هر توزیع گاوسی کروی فرض می‌شود، به این معنی که همه ابعاد دارای واریانس یکسان هستند و با یکدیگر همبستگی ندارند. این منجر به ساده‌تر و سریع‌تر شدن الگوریتم می‌شود، اما همچنین به این معنی است که خوشه‌ها فقط می‌توانند کروی شکل باشند و ممکن است توزیع‌ها پیچیده تر را با دقت پایین و یا اشتباه بازنمایی کند.

c)

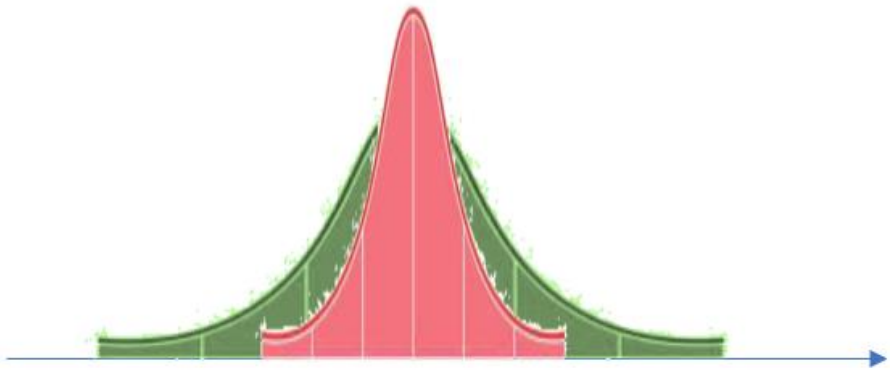
با توجه به اینکه داده‌های به شکل غیر خطی چیده شده اند الگوریتم Kmeans امکان دسته بندی به شکل صحیح را ندارد و دچار خطای بسیار زیاد خواهد شکل و نتیجه در نهایت شکل مانند شکل زیر خواهد بود.



نقاط نارنجی مرکز دسته‌ها و خط چین سبز مرز میان دسته‌ها

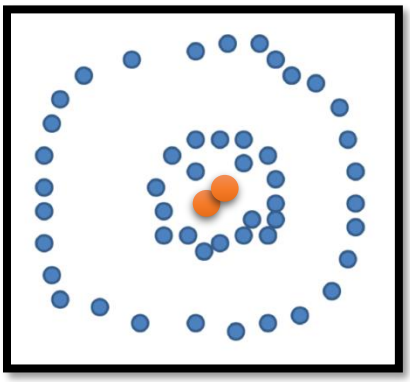
در رابطه با EM، این روش این امکان را دارد در مقابل کلاسترها با چگالی‌های متفاوت عملکرد مناسبی از خود نشان دهد اما با توجه به اینکه کلاسترهای ما overlapping هستند نتیجه خروجی ما مشابه بخش kmeans است با این تفاوت که ممکن است مرز میان دودسته کمی‌نرم تر باشد و بصورت خط نباشد و حتی برخی از داده‌ها در هر دو دست حضور داشته باشند. از این رو خود الگوریتم معیار مناسبی برای تشخیص این دو خوشه نیست اما می‌توانیم نقاطی که بعد از اجرای الگوریتم به هر دو دسته با احتمال بالا تخصیص داشته اند راکلاستر میانی و بقیه را کلاستر بیرونی اعلام کنیم و با این روش داده‌های داخلی و بیرونی را پیدا کنیم.

یک حالت دیگر نیز وجود دارد که می‌توان برای این کلاسترینگ در نظر گرفت آن هم این فرض است که بگوییم دو مرکز کلاستر در حوالی وسط تصویر قرار گیرند با این تفاوت که یکی از آن‌ها در پارامتر انحراف معیاری بزرگتری دارد در نتیجه احتمال حضور نودهای لایه بیرونی در آن کلاستر بیشتر است. توزیع احتمالاتی بر اساس فاصله به شکل زیر خواهد بود.



در نتیجه نقاط وسط احتمال بیش تری در مدلی که انحراف معیار کمتریی دارد به خود اختصاص می‌دهند و نقاط دور تر در مدلی که انحراف معیار بیشتری دارد.

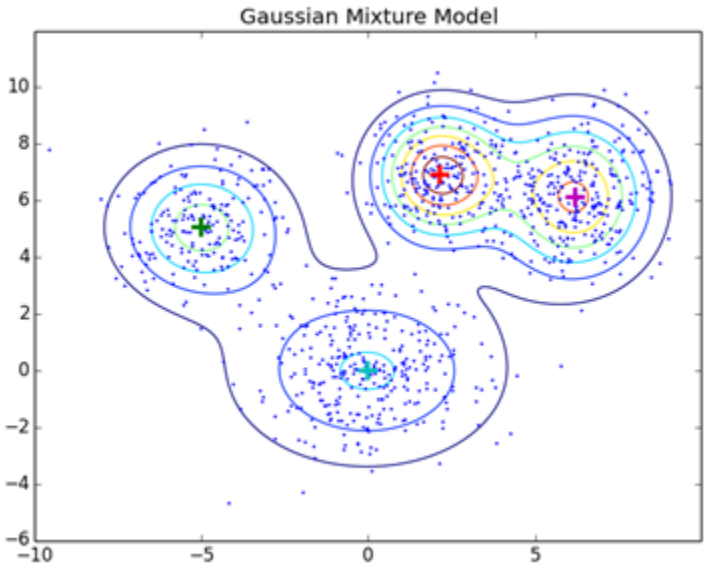
مکان حدودی مراکز دسته‌ها را می‌توانیم به شکل زیر فرض کنیم.(حتی می‌توان در یک نقطه هر دو را فرض نمود)



d)

بهترین رویکرد برای خوشه‌بندی مشتریان یک مرکز خرید برای اهداف تبلیغاتی، رویکرد مبتنی بر مدل می‌توان دانست، به‌ویژه با استفاده از Gaussian Mixture Model (GMM) می‌باشد.

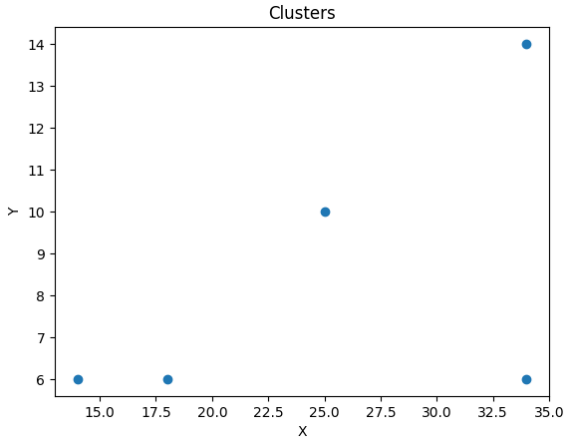
دلیل این امر این است که رویکردهای مبتنی بر مدل می‌توانند نمای ظریف‌تری از داده‌های مشتری ارائه دهند، روابط و توزیع‌های پیچیده‌ای را که ممکن است با روش‌های پارتیشن‌بندی ساده یا مبتنی بر چگالی به خوبی نمایش داده نشوند، به تصویر بکشند. GMMها به طور خاص با استفاده از توابع گوسین متفاوت سعی در مدل کردن داده‌ها دارند، که می‌تواند ساختار زیربنایی داده‌های مشتری را با دقت بیشتری بیان کند.



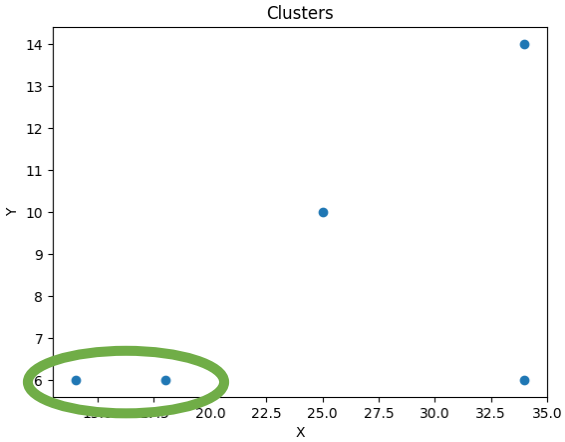
علاوه بر این، GMMها تفسیر احتمالی داده‌ها را نیز ارائه می‌دهند و به ما امکان می‌دهند احتمال تعلق مشتری به هر خوشه را تخمین بزنیم که می‌تواند برای اهداف تبلیغاتی مفید باشد. پارامترهای GMM را نیز می‌توان از روی داده‌ها تخمین زد و به ما امکان می‌دهد ویژگی‌های هر خوشه را بهتر درک کنیم و استراتژی‌های تبلیغاتی را با نیازها و ترجیحات خاص هر خوشه تطبیق دهیم.

به طور خلاصه، رویکرد مبتنی بر مدل، به‌ویژه با استفاده ازGMM ، در ترکیب با روش‌های سلسله مراتبی همانطور که در شکل می‌بینید می‌تواند قدرت و دقت بالایی برای ما ایجاد کند، زیرا می‌تواند درک دقیق‌تر و احتمالی‌تری از داده‌های مشتری ارائه دهد و برای استراتژی‌های تبلیغاتی متناسب تر است.

e)

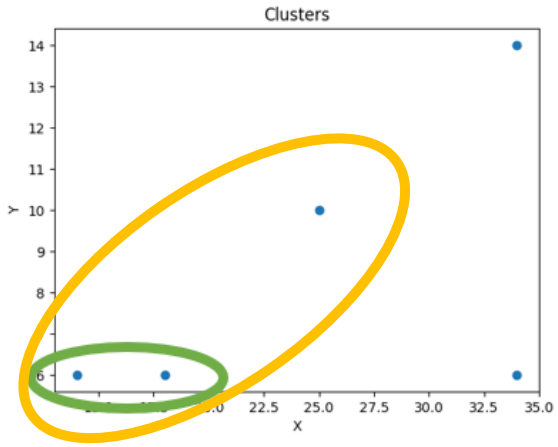


در هر مرحله کوتاه ترین فاصل را انتخاب می‌کنیم و به یک دیگر متصل می‌کنیم

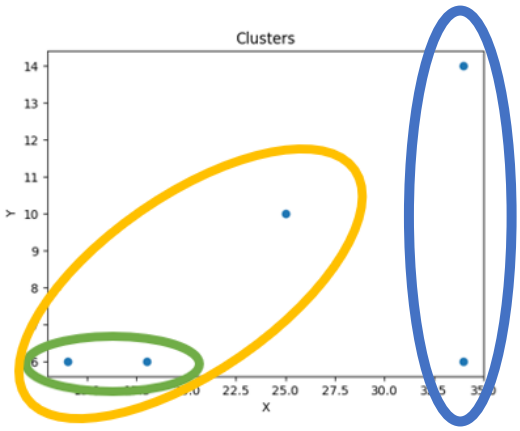


گام اول

**فاصله بین دو کلاستر برابر است با فاصله نزدیک ترین نقاط کلاسترها

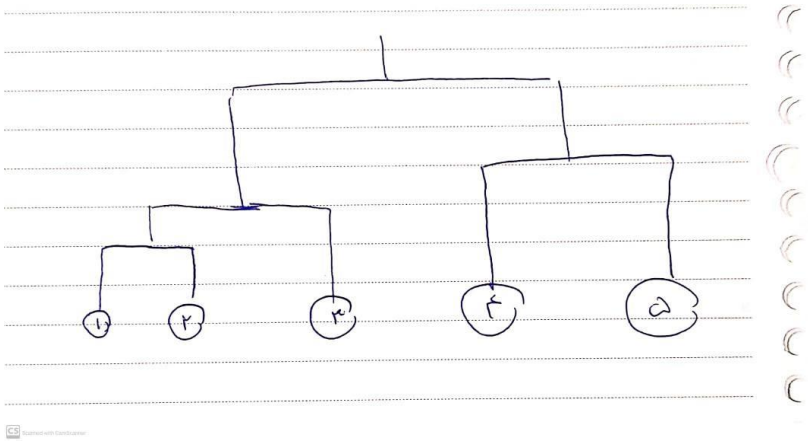


گام دوم

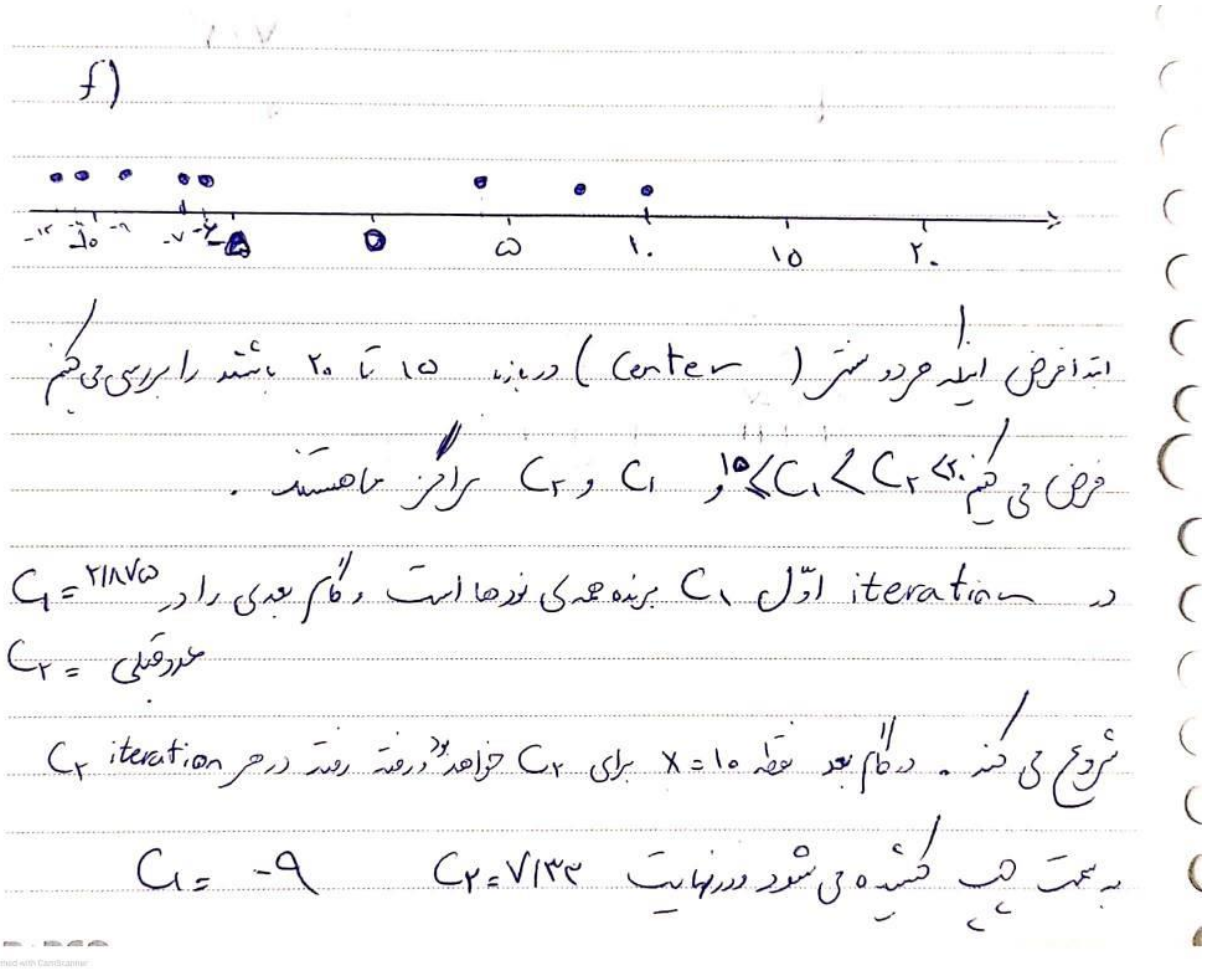


گام سوم

در گام آخر نیز همه دسته‌ها در یک دسته قرار خواهند گرفت که از نمایش آن به عنوان یک مرحله جدا صرف نظر شده است. مراحل ترکیب نودها به صورت شماتیک در زیر آورده شده است.



f)



ابتدا فرض اینکه مرز دسته (center) در بازه ۱۵ تا ۲۰ باشد را بررسی می‌کنیم

فرض می‌کنیم $15 \leq C_1 < C_2 < 20$ و C_1 و C_2 برابر ۱۵ و ۲۰ هستند.

در iteration اول C_1 برنده همه‌ی نقاط است و گام بعدی را در $C_1 = 21.75$ و C_2 = عدد قبلی

شروع می‌کند. در گام بعد نقطه $x = 10$ برای C_2 خواهد بود و همینطور در هر iteration C_2 به سمت چپ کشیده می‌شود و در نهایت $C_2 = 7.33$ $C_1 = -9$

در نتیجه در نهایت با دو مرکز به همگرایی درستی دست می یابد.

حال با دو عدد random برای C_1 و C_2 شروع می کنیم

$C_1 = -5$

$C_2 = 5$

پنج نقطه‌ی سمت چپ

↓

$C_1 = -9$

↓

همان نقطه‌های قبلی

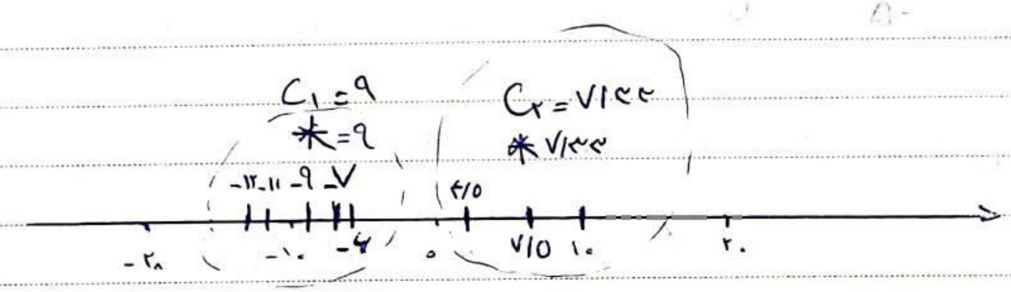
۴ نقطه‌ی سمت راست

↓

$C_2 = 7/33$

↓

همان نقاط قبلی



g)

Gaussian Mixture Model (GMM) یک مدل احتمالی است که فرض می‌کند داده‌ها توسط ترکیبی از چندین توزیع گاوسی تولید می‌شوند. هر توزیع گاوسی نشان دهنده یک خوشه است و مدل پارامترهای هر توزیع شامل میانگین، کوواریانس و وزن را تخمین می‌زند.

فراپارامترهای GMM شامل تعداد خوشه‌ها (K)، ساختار کوواریانس (مورب، کامل، کروی، و غیره) و روش اولیه سازی (به عنوان مثال تصادفی، k-means) است.

مزایای GMM عبارتند از توانایی آن در گرفتن روابط پیچیده و غیر خطی در داده‌ها، تفسیر احتمالی آن از داده‌ها، و توانایی آن برای تطبیق حالت‌های متعدد در داده‌ها. GMM همچنین از نظر ساختار کوواریانس انعطاف‌پذیر است و امکان مدل‌سازی پیچیده‌تر داده‌ها را فراهم می‌کند.

معایب GMM شامل حساسیت آن به انتخاب فراپارامترها، به ویژه تعداد خوشه‌ها و ساختار کوواریانس، و هزینه محاسباتی آن است که می‌تواند برای مجموعه داده‌های بزرگ بالا باشد.

GMM برای مجموعه‌های داده پیشنهاد می‌شود که در آن رابطه اساسی بین متغیرها پیچیده و غیرخطی است، و برای مجموعه‌های داده با حالت‌های چندگانه در داده‌ها. به ویژه در سناریوهایی که تفسیر احتمالی داده‌ها مورد نظر است مفید است. با این حال، ممکن است برای مجموعه داده‌های بزرگ یا با ابعاد بالا که در آن هزینه محاسباتی یک نگرانی است، مناسب نباشد.

Q2)

a)

علت اساسی نرمالیزیشون در روش‌های خوشه بندی مبتنی بر فاصله این است که scale کردن داده‌ها در معیار امتیاز دهی اثر مستقیم دارد و می‌تواند اینگونه برداشت شود که ویژگی با اعداد بزرگتری، اهمیت بیشتری دارد برای مثال ویژگی CHOL در دیتاست ما در بازه (۹٫۶۷, ۱٫۴۳) قرار دارد اما ویژگی ALB در بازه (۸۲٫۲, ۱۴٫۹) همان طور که مشاهده می‌کنید اندازه این دو ویژگی نسبتا حدود یک به ده دارد و این باعث می‌گردد که ALB صرفا بخاطر scale بزرگ تر، وزن ده برابری نسبت به CHOL داشته باشد در صورتی که ممکن است اهمیت این دو فیچر برابر باشند.

در این بخش علاوه بر اصلاحات گفته شده مقادیر null نیز با میانگین ستون پر گردیده است.

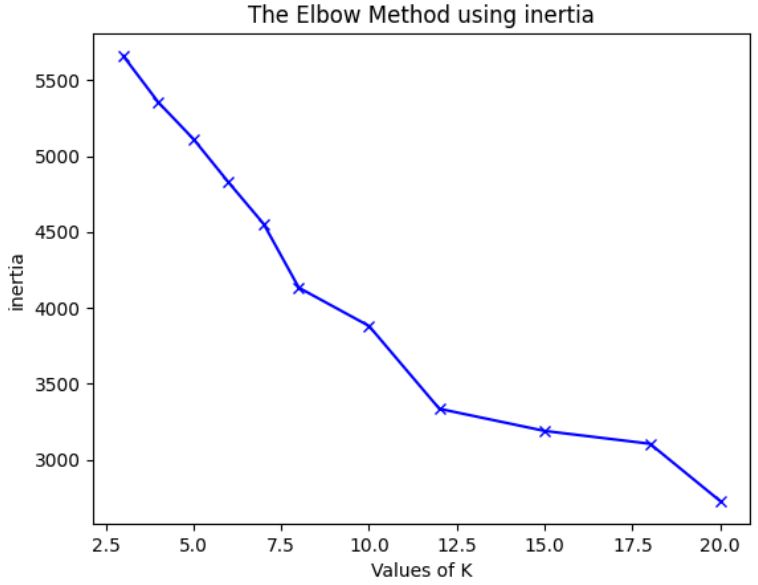
برای نرمالسازی نیز از فاصله تا میانگین تقسیم بر انحراف معیار استفاده شده است.

b)

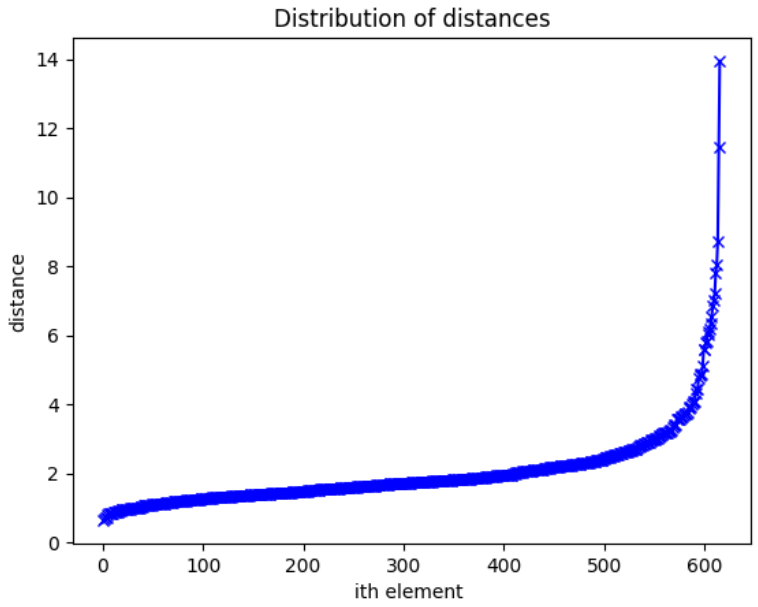
در این بخش تابع مورد نظر پیاده سازی گردیده است 😊

c)

در این بخش ابتدا با استفاده از روش Elbow مقدار k مناسب برای Kmeans را استخراج می‌کنیم. با توجه به تکرارهای پیاپی می‌توانی k = 12 یا حدود آن را مناسب در نظر بگیریم.



سپس فواصل موجود تا مرکز دسته مربوط به هر داده را صورت می‌کنیم.



در این جا بصورت چشمی حد آستانه فاصله را ۴ قرار می‌دهیم و بیشتر از آن را داده پرت در نظر می‌گیریم (لازم به ذکر است که در اینجا داده‌های ورودی نرمال شده اند)

با این کار میزان ۴.۵ درصد از داده‌ها به عنوان خطا شناسایی می‌شوند (در تکرارهای متفاوت با توجه به نتایج متفاوت الگوریتم Kmeans به حدود ۶ درصد نیز رسید)

d)

با استفاده از کتابخانه sklearn انجام گرفته است

e)

در این بخش ابتدا لیبل‌های ۱ تا ۴ را به ۱ و ۲ تبدیل می‌کنیم (همانطور که در صورت سوال آمده است).

در گام بعدی Kmeans را اجرا می‌کنیم و به دو کلاستر دست پیدا می‌کنیم. مهم ترین مساله در این بخش این است که چه لیبلی برای این کلاسترها تعیین کنیم، برای این کار دو را برای لیبل گذاری پیشنهاد شده است که با way1 و way2 در کد به آن‌ها اشاره گردیده است.

Way1: لیبل یک کلاستر حاصل از رای گیری بین نودهایی است که در بخش train داخل آن حضور دارند. مشکل این روش این است که داده‌های ما عموما لیبل صفر دارند و در هر دو دسته پیروز می‌شوند.

Way2: این روش دیدی برعکس دارد و می‌گوید اکثریت هر دسته در کدام خوشه قرار دارند برای مثال اگر در کل داده‌های تست ۲۰ داده از دسته ی A وجود داشته باشد اگر ۱۲ داده در یک خوشه از A دیده شود این کلاستر A می‌شود. این روش مشکلات خاص خود را دارد مثل اینکه ممکن است دو label برای یک کلاستر انتخاب شود اما در این دیتاست باعث می‌شود از اینکه هر دو کلاستر لیبل صفر را به خود بگیرند جلوگیری کند. در بخش بعدی نیز تا حدودی این کار را می‌کند.

در ادامه نتایج بدست آمده از هر روش را درج خواهیم نمود.

```
way1 accuracy : 0.8983050847457628
way2 accuracy : 0.4576271186440678
```

```
way1 entropy : 0.4320447046064277
way2 entropy : 0.4320447046064277
```

```
way1 purity : 0.9104477611940298
way2 purity : 0.4626865671641791
```

از نظر معیارها در همه حالات روش اول پیروز است اما دلیل آن این است که نسبت دسته صفر خیلی خیلی بیشتر از لیبل‌های دیگر یا دسته‌های دیگر است.

** از آنجا که ممکن است در حین مطالعه در ترجمه عبارات ابهام وجود داشته باشد به بخش زیر توجه کنید

دسته = label =class

خوشه = cluster

f)

در این بخش نیز مشابه بخش قبل عمل شده است و برای هر دو روش مقادیر در پایین آورده شده است.

```
way1 accuracy : 0.8983050847457628
way2 accuracy : 0.5932203389830508
```

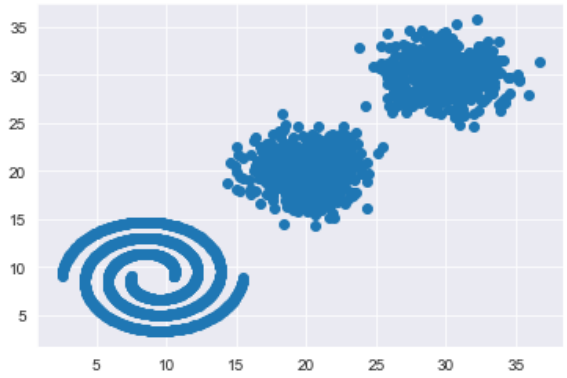
```
way1 entropy : 0.503423806390791
way2 entropy : 0.503423806390791
```

```
way1 purity : 0.9104477611940298
way2 purity : 0.5991471215351812
```

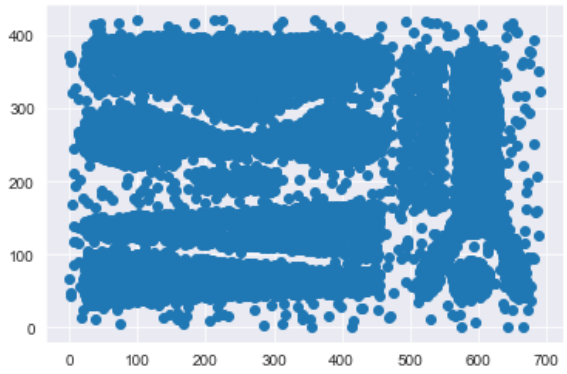
Q3)

a)

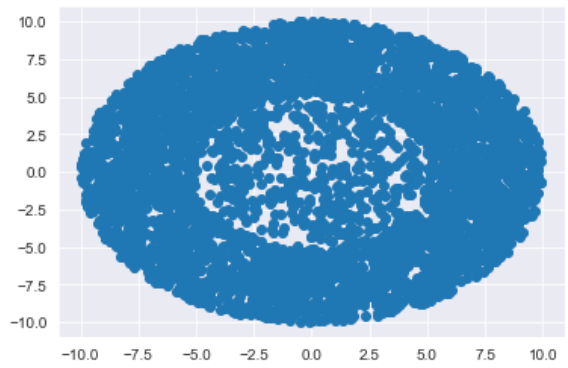
در دیتاست ۱ به وضوح ۳ کلاستر موجود می‌باشد (بخش مارپیچ را می‌توان دو تا در نظر گرفت که در اینصورت ۴ کلاستر خواهیم داشت)



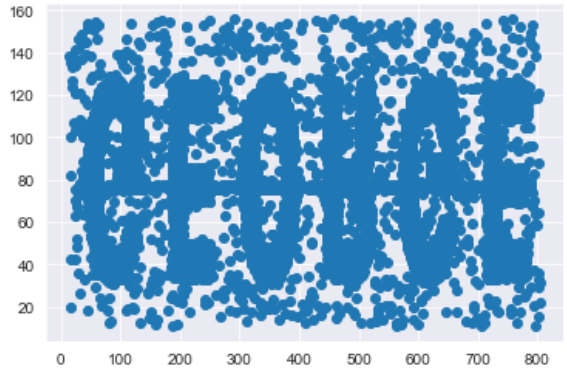
در دیتاست ۲ ، کلاسترها به خوبی جدا نشده اند اما بصورت چشمی ۸ بخش مجزا دیده می‌شود(۵ تا در سمت چپ و ۳ تا در سمت راست)



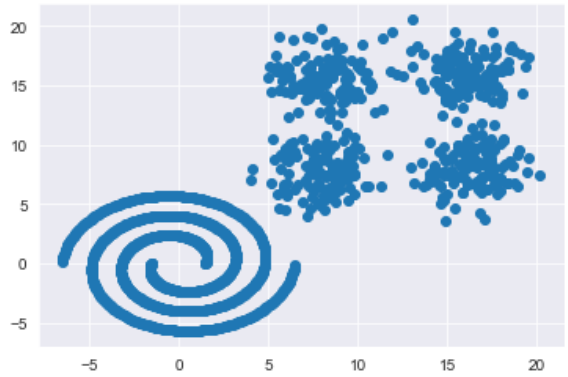
در دیتاست ۳ یک کاستر بیرونی مشاهده می‌شود و بقیه نقاط در بخش کم تراکم تر هستند، می‌توان فرض کرد یک کلاستر بیرونی و بقیه نویز هستند و یا اینکه بگوییم کلا دو کلاستر دارم.



۶ کلاستر قابل مشاهده اند اما بدلیل اینکه همگی به هم متصل هستند به احتمال زیاد DBscan امکان تشخیص اینها را نداشته باشد.



دیتاست ۵ نیزبه وضوح دارای ۵ کلاستر یا خوشه می‌باشد.(با این فرض که بخش مارپیچ یک کلاستر می‌باشد)



b)

در این بخش DBscan پیاده سازی گردیده است که نتایج آن در بخش بعدی قابل مشاهده می‌باشد. لازم به ذکر است که معیار فاصله در تمام این سوال فاصله اقلیدسی می‌باشد.

```
DBSCAN(DB, distFunc, eps, minPts) {
    C := 0                                                  /* Cluster counter */
    for each point P in database DB {
        if label(P) ≠ undefined then continue               /* Previously processed in inner loop */
        Neighbors N := RangeQuery(DB, distFunc, P, eps)     /* Find neighbors */
        if |N| < minPts then {                              /* Density check */
            label(P) := Noise                               /* Label as Noise */
            continue
        }
        C := C + 1                                          /* next cluster label */
        label(P) := C                                       /* Label initial point */
        SeedSet S := N \ {P}                                /* Neighbors to expand */
        for each point Q in S {                             /* Process every seed point Q */
            if label(Q) = Noise then label(Q) := C          /* Change Noise to border point */
            if label(Q) ≠ undefined then continue           /* Previously processed (e.g., border
point) */
            label(Q) := C                                   /* Label neighbor */
            Neighbors N := RangeQuery(DB, distFunc, Q, eps) /* Find neighbors */
            if |N| ≥ minPts then {                          /* Density check (if Q is a core point) */
                S := S ∪ N                                  /* Add new neighbors to seed set */
            }
        }
    }
}
```

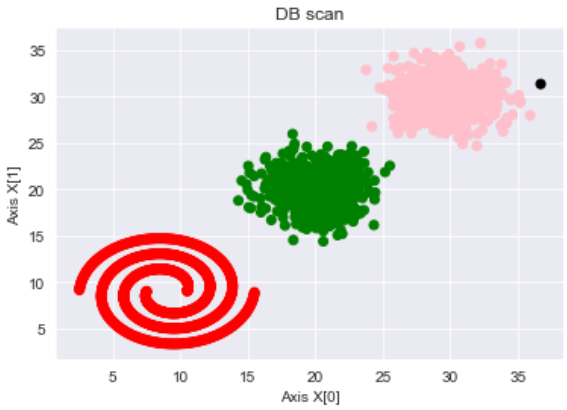
شبه کد مورد استفاده برای پیاده سازی

c)

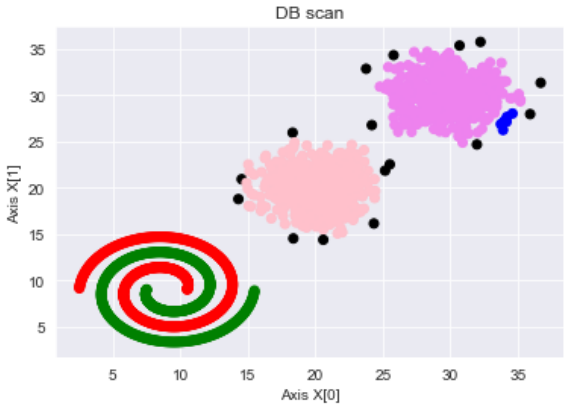
در این بخش برای هر دیتاست حالات مناسب برای دسته بندی را محاسبه کرده و نشان دادیم

** در همه حالات سیاه به معنی داده پرت است

دیتاست ۱

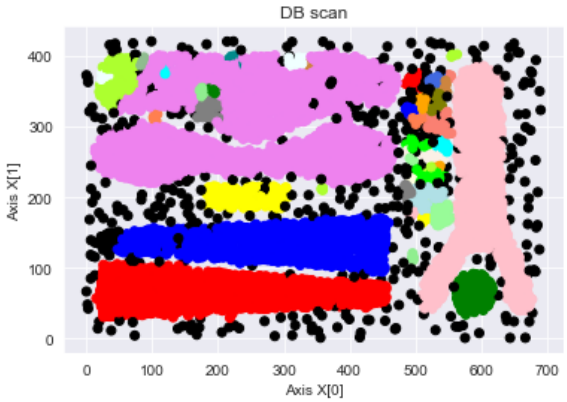


Epsilon = 2, Minpoints = 3 (3 clusters)



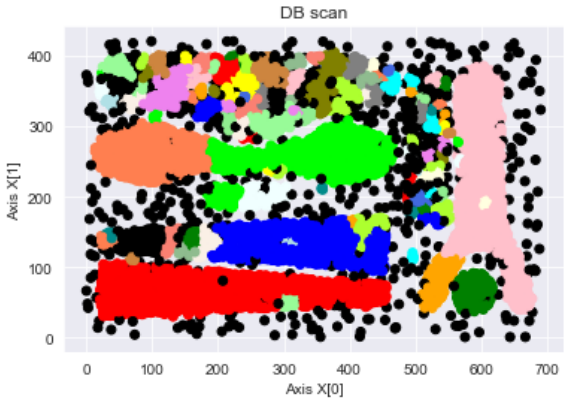
Epsilon = 1, Minpoints = 3 (5 clusters)

دیتاست ۲



Epsilon = 7, Minpoints = 3 (47 clusters)

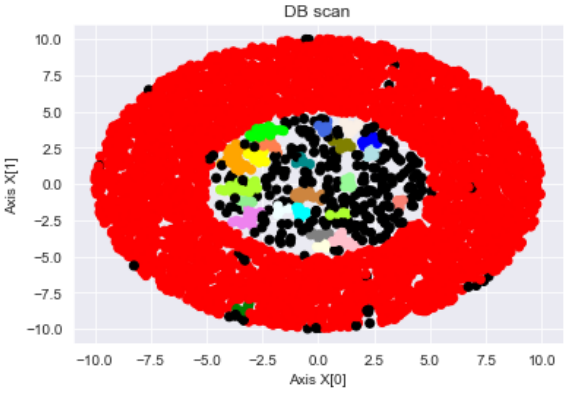
دیتاست ۲



Epsilon = 6, Minpoints = 3 (130 clusters)

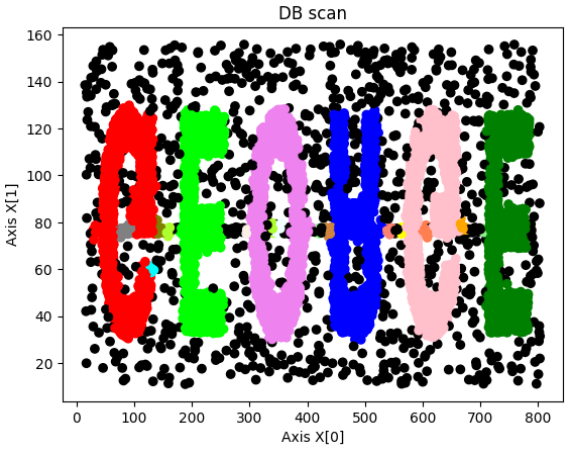
دیتاست ۳

علت این اتفاق در دیتاست ۳این است که چگالی داده‌ها در دو دسته اصلی یکسان نیستند و الگوریتم در یکی از ناحیه‌ها دچار اشکال خواهد شد که در تصویر زیر کاملا مشهود است که دسته میانی به شکل نامناسبی دسته بندی شده است.



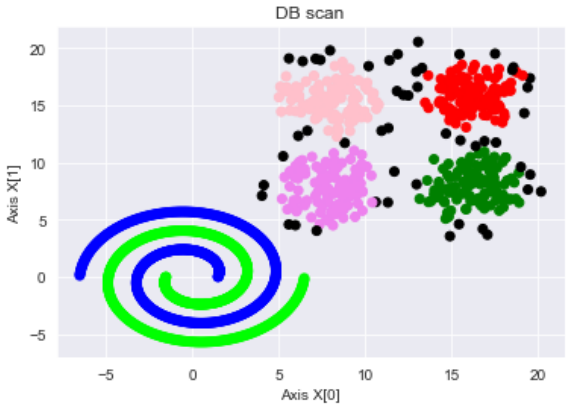
Epsilon = 0.4, Minpoints = 3 (26 clusters)

دیتاست ۴



Epsilon = 4, Minpoints = 6 (21 clusters)

دیتاست ۵



Epsilon = 1, Minpoints = 5 (6 clusters)

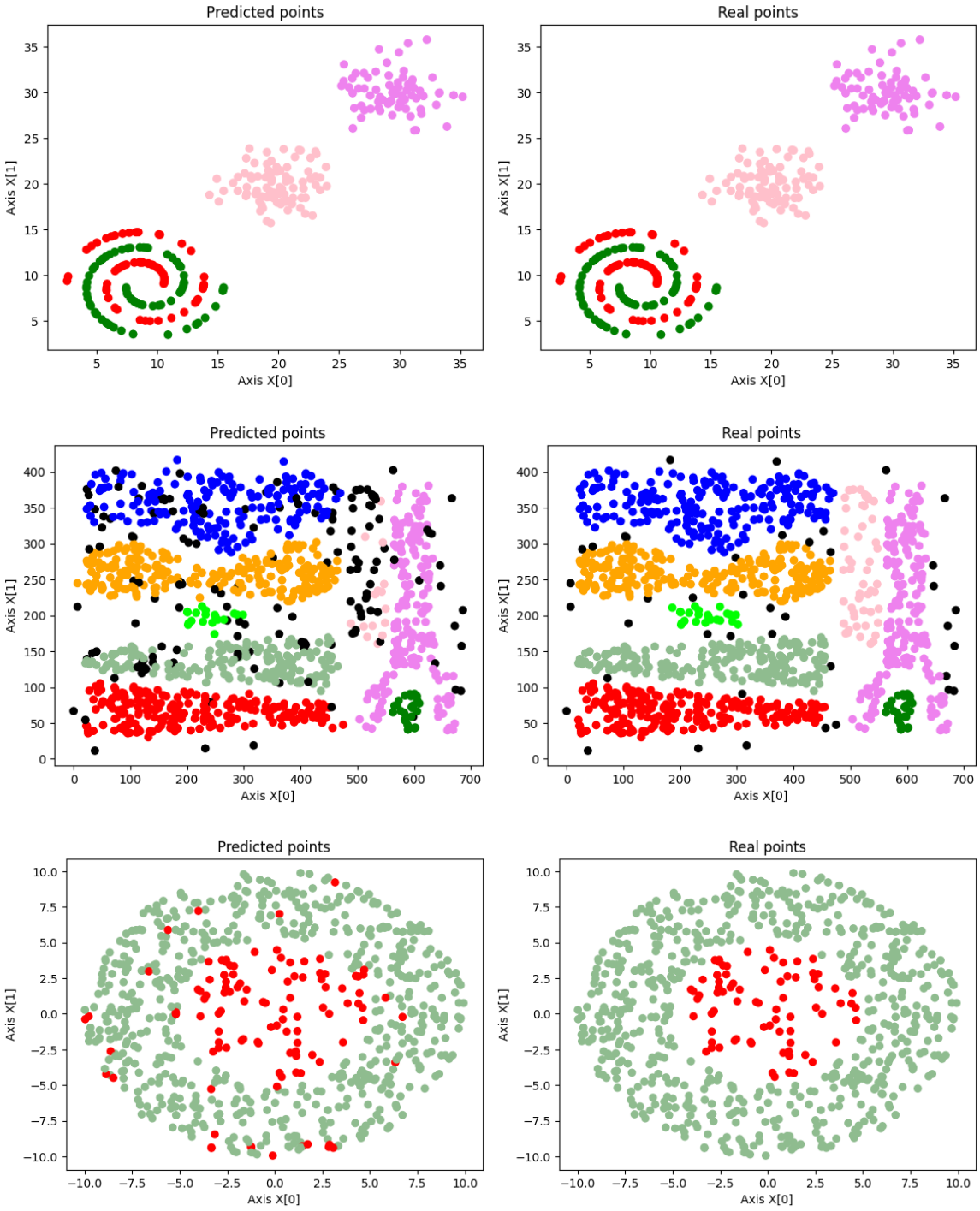
d)

الگوریتم DBscan بعد از اجرا شدن به هر یک از کلاسترهای ما یک ID می‌دهد از آنجا که این ID برای مقایسه مناسب نیست ابتدا با توجه به داده‌های train، label هر کلاستر را مشخص می‌کنیم. در اینجا ممکن است ۲۰ کلاستر و ۵ label وجود داشته باشد و این مورد برای ما مشکلی ایجاد نمی‌کند و ممکن است چند کلاستر label یکسانی داشته باشند(این مرحله در کد با نام convert آورده شده است). پس از آن برای هر داده ورودی لیبل نزدیک ترین کلاستر را بررسی می‌کنیم و براساس آن label داده ورودی را تعیین می‌کنیم.

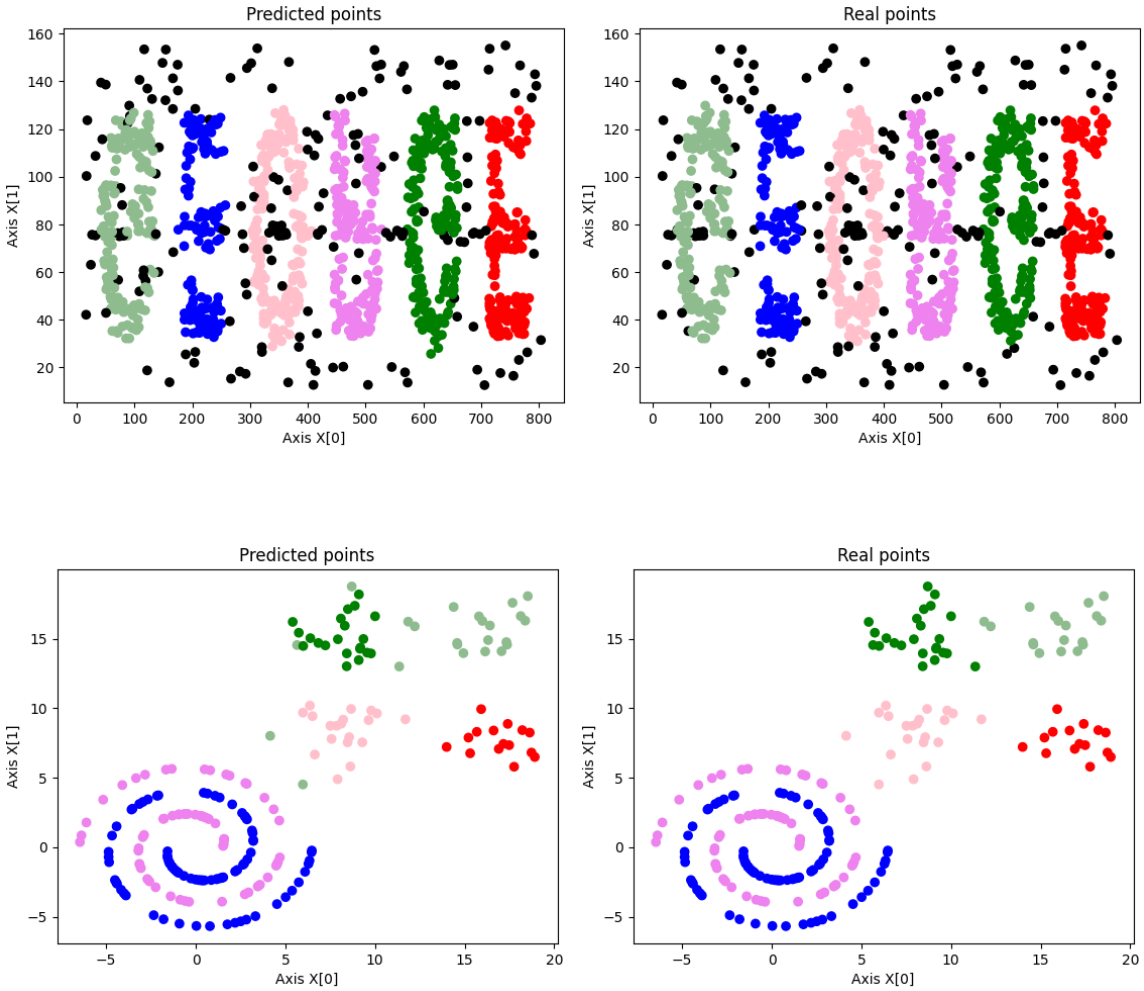
در این بخش ابتدا به هر یک از کلاسترها یک لیبل می‌دهیم، سپس برای هر نقطه نزدیک ترین دسته ای که یافت می‌شود را پیدا می‌کنیم. در این بخش یک مرحله convert وجود دارد که لیبل‌هایی که در بخش کلاسترینگ وجود دارد را با استفاده از داده‌های train بروز رسانی می‌کند. به بیان دیگر شاید کلاسترینگ ما ۲۰ کلاستر شناسایی کند ولی ما در واقع ۵ دسته داشته ایم این مرحله convert مشخص می‌کند که هر کلاستر به کدام یک از ۵ دسته اصلی بدل خواهند شد.

e)

name	Epsilon	Minpoint	Accuracy
Dataset1	1	3	100%
Dataset2	6	3	88%
Dataset3	0.4	5	95%
Dataset4	4	6	97.7%
Dataset5	1	6	97.7%



علت پخش شدن نقطه‌های قرمز داخل ناحیه سبز این است که در این مدل داده‌های پرت را جزو عمدتا جزو دسته قرمز تشخیص داده برای همین اگر داده پرت هر جا باشه بهش لیبل قرمز می‌دهد این کار در مجموع باعث بهبود عملکرد می‌شود اما نقطه‌های قرمز پراکنده ایجاد می‌کند.



دراین شکل نیز داده‌های پرت جزو دسته سبز فرض شدن و این موضوع باعث این می‌شود که آن دو نقطه میانی سبز کم رنگ تشخیص داده شوند زیرا داده‌های پرت در این شکل عموما به دست سبز کمرنگ تعلق داشته اند.

f)

3NN	Accuracy	5NN	Accuracy
Dataset1	100%	Dataset1	100%
Dataset2	98.6%	Dataset2	98.0%
Dataset3	100%	Dataset3	99.8%
Dataset4	97.7%	Dataset4	97.1
Dataset5	100%	Dataset5	99%

در کمال تعجب باید گفت که عملکرد KNNها در همه حالات بهتر مساوری DBscan بود و در نتیجه KNN را می‌توان پیروز این مقایسه دانست. علت این موضوع در این است که اگر DBscab یک عنصر را به درستی به یک دسته مپ کند KNN نیز قطعا این کار را می‌کند زیرا تعداد همسایگی‌هایی که برای عضو شدن در یک مجموعه وجود دارد برای اینکه KNN نیز همان تصمیم گیری کند عموما کافی است. نقطه تمایز KNN در مرزهای بین کلاسترها است، DBscan دو مجموعه داده با لیبل‌های متفاوت ولی نزدیک از نظر فاصله اقلیدسی را یکی فرض می‌کند. این در شرایطی است که KNN به راحتی می‌تواند این دو را از یک دیگر جدا کند و فقط در مرز خطا داشته باشد ولی DBscan خطایش را در همه ناحیه اشتباه تصمیم گرفته شده انتشار می‌دهد و به نظر من اصلی ترین تفاوت این دو روش همین نکته است.

Q4)

a)

Actions: حرکت به سمت بالا ، پایین ، چپ و راست

Observation: جدول موقعیت المان‌ها 4 * 4

Reward: در آخرین گام درصورت رسیدن به هدف ۱ و در غیر این صورت ۰

b)

در ۱۰۰۰ بار اجرای رندوم مرحله در کمتر از ۲ درصد به موفقیت عامل دست یافت.

```
Results after 1000 episodes:
number of wins: 19
```

c & d)

در این بخش مساله را در دو حالت آموزش داده ایم

حالت اول

حالت ساده که در آن لغزندگی وجود ندارد در این حالت مدل به راحتی به دقت کامل رسید و در همه تکرار ها مساله را حل نمود.

```
Results after 1000 episodes:
number of wins: 1000
```

هایپر پارامترها در این بخش به این شکل می‌باشد.

```
3  # Hyper parameters
4  alpha = 0.1
5  gamma = 0.6
6  epsilon = 0.3
```

جدول Q

	0	1	2	3
0	0.046656	0.07776	0.07776	0.046656
1	0.046656	0.00000	0.12960	0.077760
2	0.077760	0.21600	0.07776	0.129600
3	0.129600	0.00000	0.07770	0.077421
4	0.077760	0.12960	0.00000	0.046656
5	0.000000	0.00000	0.00000	0.000000
6	0.000000	0.36000	0.00000	0.129600
7	0.000000	0.00000	0.00000	0.000000
8	0.129600	0.00000	0.21600	0.077760
9	0.129600	0.36000	0.36000	0.000000
10	0.216000	0.60000	0.00000	0.216000
11	0.000000	0.00000	0.00000	0.000000
12	0.000000	0.00000	0.00000	0.000000
13	0.000000	0.36000	0.60000	0.216000
14	0.360000	0.60000	1.00000	0.360000
15	0.000000	0.00000	0.00000	0.000000

ستون‌های از ۰ تا ۳ به ترتیب عبارت اند از چپ، پایین، راست و بالا می‌باشد

در رابطه با سیاست نیز بهترین عمل ممکن برای در یک استیت ذکر شده است.

```
state 0: 1
state 1: 2
state 2: 1
state 3: 0
state 4: 1
state 5: Endpoint
state 6: 1
state 7: Endpoint
state 8: 2
state 9: 1
state 10: 1
state 11: Endpoint
state 12: Endpoint
state 13: 2
state 14: 2
state 15: Endpoint
```

حالت دوم

در این حالت فرض لغزندگی را نیز فعال کردیم و مساله کمی دشوار تر می‌باشد.

هایپر پارامترهای به کار گرفته به شکل زیر هستند.

```
# Hyper parameters
alpha = 0.2
gamma = 0.95
epsilon = 0.2
```

```
Results after 1000 episodes:
number of wins: 736
```

جدول Q

	0	1	2	3
0	0.166723	0.162360	0.157048	0.154783
1	0.078293	0.059819	0.111299	0.149586
2	0.137336	0.131955	0.127784	0.128328
3	0.070725	0.102229	0.101622	0.119738
4	0.176360	0.163867	0.152116	0.072161
5	0.000000	0.000000	0.000000	0.000000
6	0.093087	0.013301	0.084783	0.032350
7	0.000000	0.000000	0.000000	0.000000
8	0.143296	0.182161	0.214202	0.261339
9	0.249367	0.375900	0.192723	0.247379
10	0.365946	0.245798	0.319713	0.165102
11	0.000000	0.000000	0.000000	0.000000
12	0.000000	0.000000	0.000000	0.000000
13	0.264134	0.352144	0.503263	0.365693
14	0.578252	0.756149	0.618104	0.603401
15	0.000000	0.000000	0.000000	0.000000

Policy

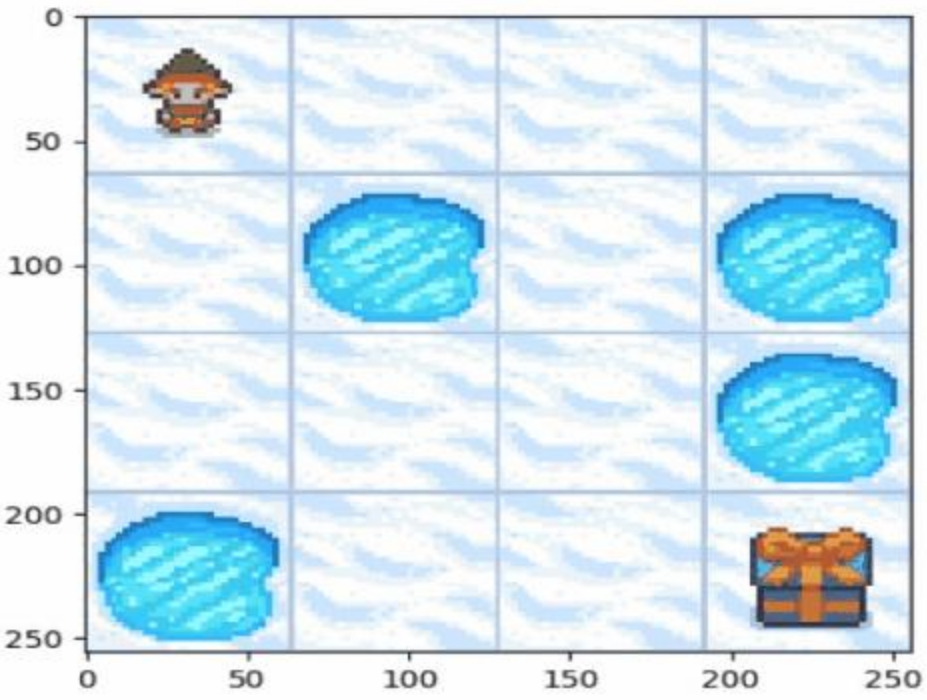
```
The policy
state 0: 0
state 1: 3
state 2: 0
state 3: 3
state 4: 0
state 5: Endpoint
state 6: 0
state 7: Endpoint
state 8: 3
state 9: 1
state 10: 0
state 11: Endpoint
state 12: Endpoint
state 13: 2
state 14: 1
state 15: Endpoint
```

e)

همان طور که از نتایج نیز پیداست یادگیری تقویتی توانسته است در حدود ۷۳.۶ درصد موفقیت در حل مساله ای که قطعیت ندارد به جواب دست پیدا کند و این در صورتی است که تصمیم گیری تصادفی به کمتر از ۲ درصد دست یافته است.

f)

گیف مربوطه در فایل زیپ شده موجود می‌باشد.


0
50
100
150
200
250
0
50
100
150
200
250