

به نام خدا

تمرین سری 5 کلاسترینگ

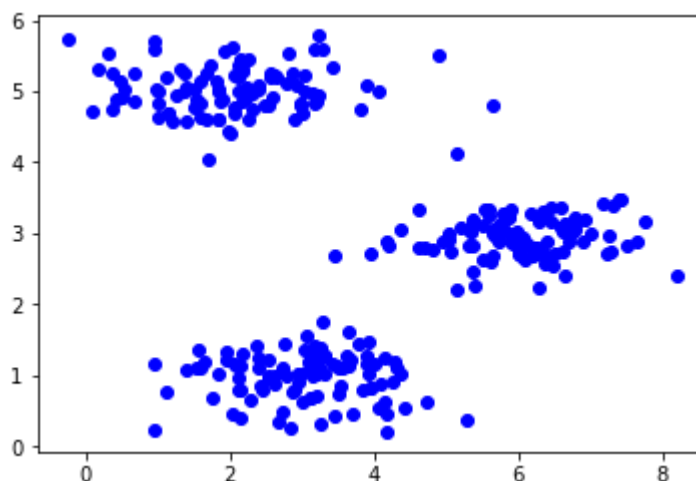
حسین غلامی 97123021

در ابتدا به دلیل این که فایل اطلاعات `.mat` است ، توسط `matlab` باز کرده ، ماتریس (جدول) را در `excel` کپی کرده و با فرمت `CSV` ذخیره میکنیم. در پوشه قرار گرفت.

سوال 1

الف (در این بخش رسم داده ها را مد نظر دارد که به شرح زیر است.

همانطور که مشاهده میشود ، به سه بخش تقسیم میشوند:



ب) در این بخش از ما الگوریتم `kmeans` را میخواهد که بدون کتابخانه ، پیاده سازی کنیم.(نقطه شروع رندم)

برای این کار ابتدا : 3 نقطه رندم به عنوان `centroid` در نظر میگیریم

```
#generate random centroid and new one  
s = [np.array(df.sample()) for x in range(3)]
```

و `centroid` جدید را مقدار دهی اولیه میکنیم حال وارد الگوریتم میشویم ، برای این کار از یک حلقه `while` که شرط توقف آن تغییر نکردن `CS` ها باشد (و همچنین `ittration`) پیاده سازی میکنیم

```
#calculate distant ,distant of each data ,from him centroid
```

```
l1=df.agg(d1,axis=1)
```

```
l2=df.agg(d2,axis=1)
```

```
l3=df.agg(d3,axis=1)
```

حال فاصله همه ی نقاط را تا CS ها محاسبه میکنیم

```
#clustering data
```

```
for i in range(len(df)):
```

```
    a=[l1[i],l2[i],l3[i]]
```

```
    cat=a.index(min(a))
```

```
    if cat==0:
```

```
        c1.append(df.values[i])
```

```
    elif cat==1:
```

```
        c2.append(df.values[i])
```

```
    else:#cat=3
```

```
        c3.append(df.values[i])
```

و کلاسترینگ اولیه را انجام میدهیم

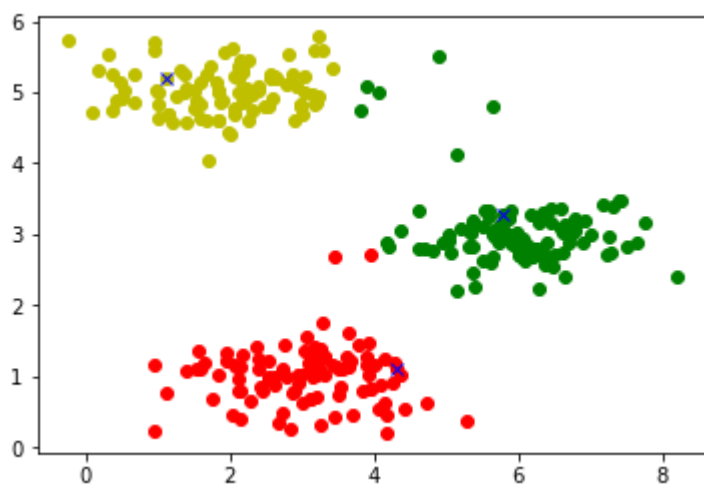
حال مرکز جدید را آپ دیت کرده و مجدد همین روند را اجرا میکنیم

```
csn[0]=np.array([[np.mean(c1p[:,0]),np.mean(c1p[:,1])]])
```

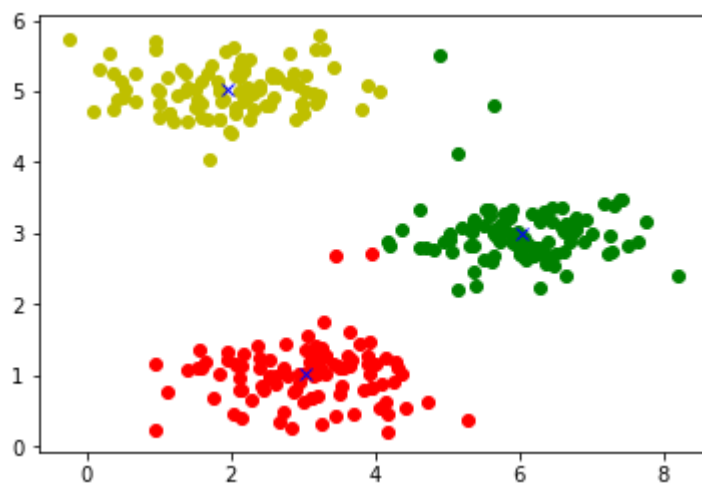
```
csn[1]=np.array([[np.mean(c2p[:,0]),np.mean(c2p[:,1])]])
```

```
csn[2]=np.array([[np.mean(c3p[:,0]),np.mean(c3p[:,1])]])
```

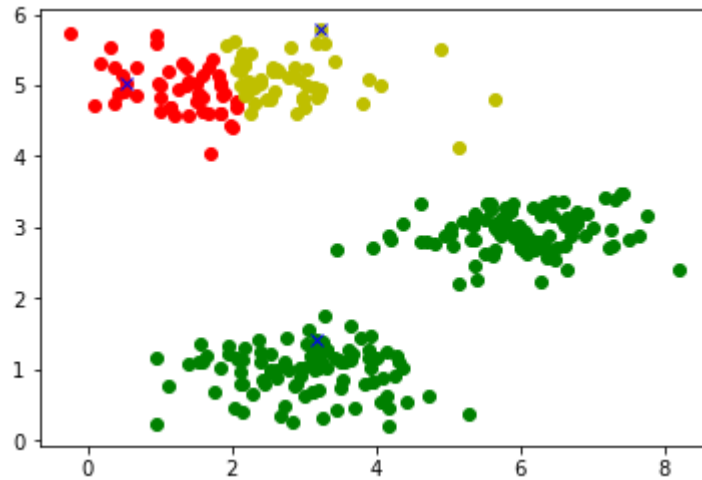
و در پایان هر بار رسم میکنیم برای مثال :



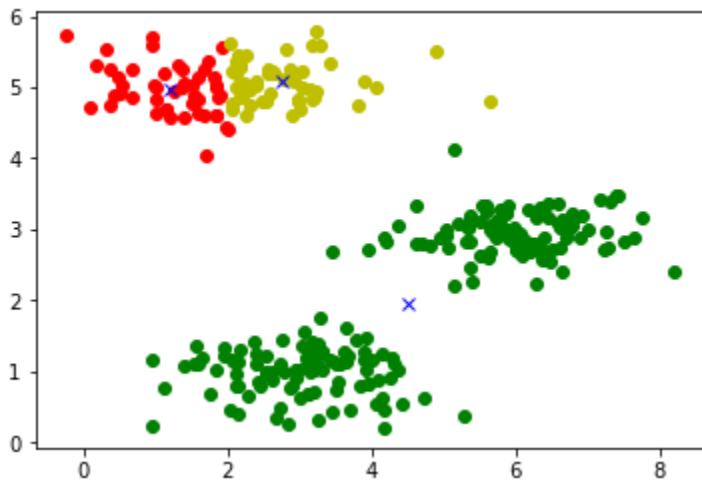
که در 3 مرحله به شکل زیر ختم شد:



ولی نقاط همواره به شکل فوق تقسیم نمیشود برای مثال :



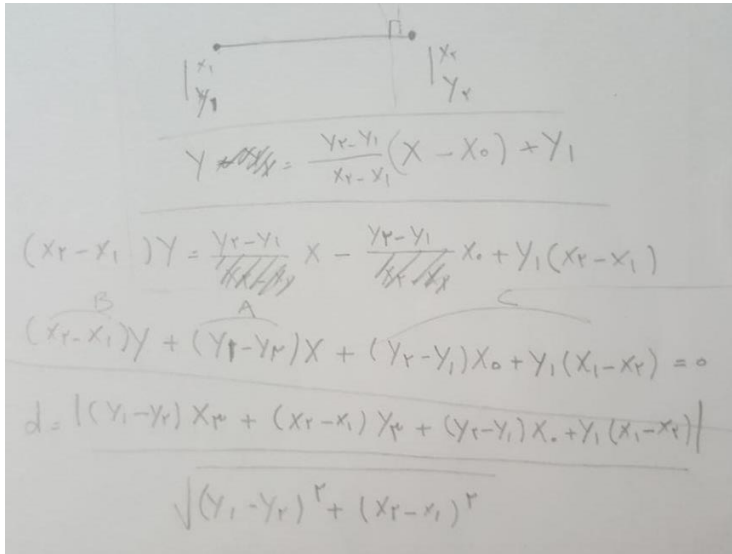
که با 5 بار اجرا به شکل زیر تبدیل شد



برای حل مشکل به پیاده سازی بخش پ ، پرداخته شد

پ) برای رفع مشکل فوق اگر فاصله دور ترین نقاط را از هم محاسبه کنیم ، میتوانیم مشکل را حل کنیم

به این شکل که ابتدا نقطه ای رندم انتخاب نموده ، فاصله ی همه ی نقاط را تا آن حساب میکنیم دور ترین نقطه را نقطه شروع دوم در نظر میگیریم و آن را حذف میکنیم ، - تا اینجا 2 نقطه داریم ، با نقطه فوق خطی تشکیل داده و فاصله همه نقاط را تا آن خط محاسبه میکنیم ، دور ترین نقطه را نقطه سوم می نامیم ، و الگوریتم قبلی را شروع میکنیم



$$Y - Y_1 = \frac{Y_r - Y_1}{X_r - X_1} (X - X_1) + Y_1$$

$$(X_r - X_1) Y = \frac{Y_r - Y_1}{1} X - \frac{Y_r - Y_1}{1} X_1 + Y_1 (X_r - X_1)$$

$$(X_r - X_1) Y + (Y_1 - Y_r) X + (Y_r - Y_1) X_1 + Y_1 (X_1 - X_r) = 0$$

$$d = \frac{|(Y_1 - Y_r) X_r + (X_r - X_1) Y_r + (Y_r - Y_1) X_1 + Y_1 (X_1 - X_r)|}{\sqrt{(Y_1 - Y_r)^2 + (X_r - X_1)^2}}$$

محاسبه تابع برای فاصله نقطه از خط :

$$PH = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

با توجه به این که مخرج کسر ثابت است :

تنها صورت پیاده سازی میشود.

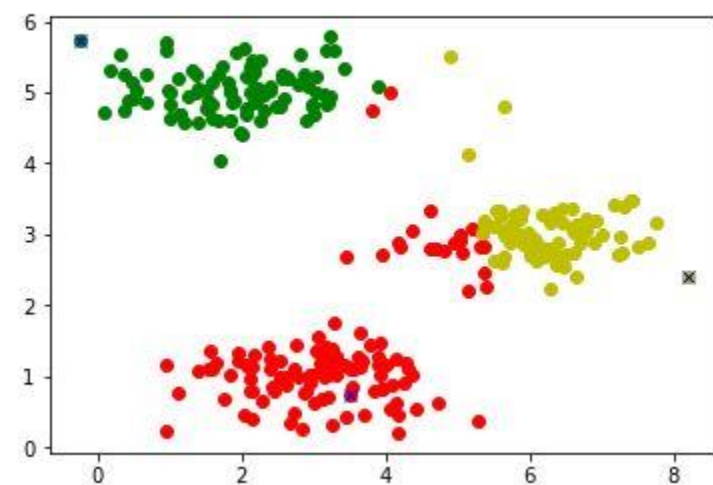
```
cs=[np.array(df.sample())]
cs1_index=df.agg(d1,axis=1).idxmax()
cs1=df[cs1_index:cs1_index+1]
dfe=df.drop(cs1_index)
cs2_index=dfe.agg(d,axis=1).idxmax()
cs2=df[int(cs2_index):int(cs2_index)+1]

#nnnend centroid

def distant_nod_from_line(nod1,nod2,nod3):
    return ( abs( (float(nod1.y)-nod2[1])*nod3.x + (nod2[0]-float(nod1.x))*nod3.x ) )

d = lambda s: distant_nod_from_line(cs1,cs[0][0],s)
```

algorithm has started



که برای مثال:

نقطه ای که در ناحیه قرمز است به عنوان
نقطه رندم انتخاب شده، نقطه ای که در ناحیه سبز
است به عنوان دور ترین نقطه انتخاب شده
و نقطه ای که در ناحیه زرد است به عنوان
دورترین نقطه نسب به خط است.
بقیه الگوریتم مشابه قبل است.

```
from sklearn.cluster import KMeans
```

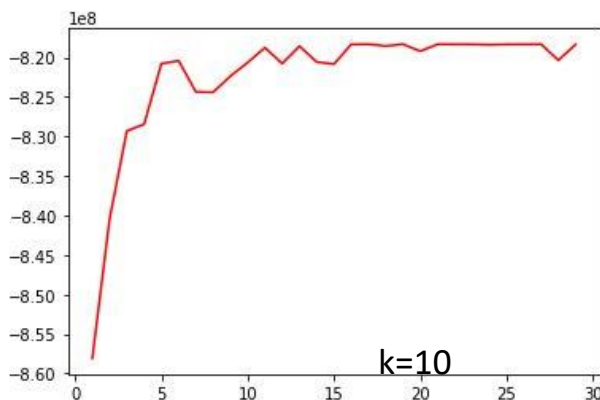
در این بخش به دلیل تعدد ابعاد اطلاعات ، و تنوع تعداد کلاسترینگ از استفاده شد .

الف) تابع هندلر kmeans به شکل زیر است که توضیحات آن نوشته شد:

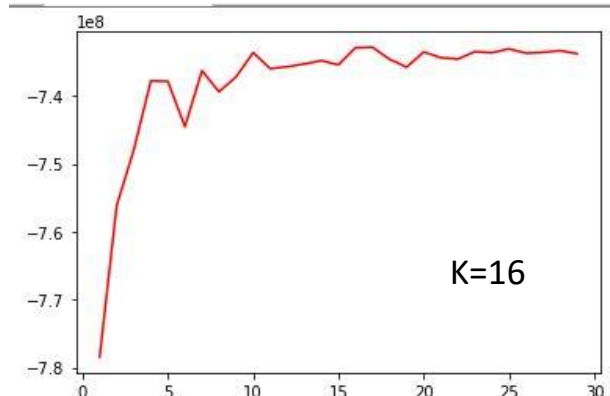
```
kmeans = KMeans(
    n_clusters=i,      #tedad cluster
    init='k-means++',  #doortarin noghgte,
    #ndarray ham baraye delkhah (n_clusters, n_features)
    n_init=10,          #tedad bar hayi ke algoritm bayad ejra shavad
    max_iter=300,       #maximom tekar algoritm
    tol=0.0001,         #tolerance baraye converg shodan
    precompute_distances='auto', #if active (faster but takes more memory)
    algorithm='full'
)
```

با مقدار دهی $n_init=10$ بار اجرا شده و بهترین نتیجه را اعلام میکند ، $init$ وقتی به صورت $k\text{-means++}$ است مقدار دهی اولیه با همان معیار دورترین عمل میکند. Max_itter تعداد دفعات اجرای الگوریتم را محدود میکند

ب) در این تابع تابعی تحت عنوان $score$ دارد که مجموع فواصل را اعلام میکند ، با توجه به آنچه مد نظر صورت سوال است ، میزان تغییرات $score$ را به برای هر بار $iteration$ میخواند پس $max_iteration$ را هر بار یک واحد اضافه نموده و منحنی را رسم میکنیم:

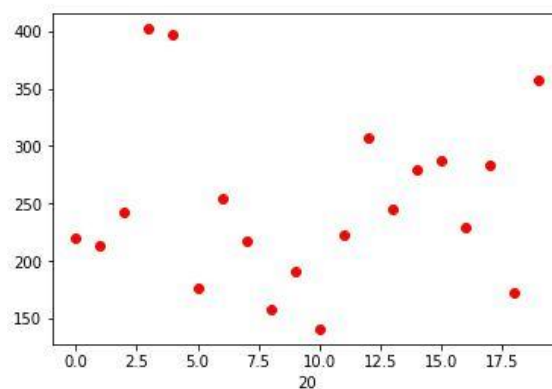
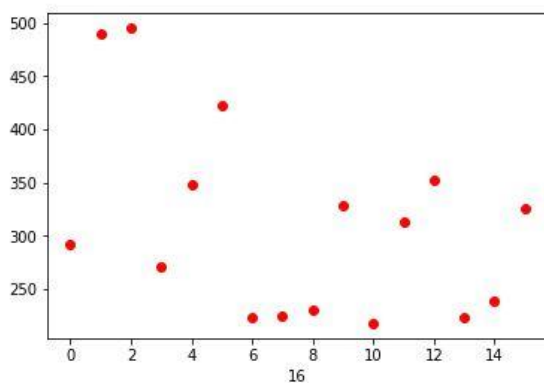
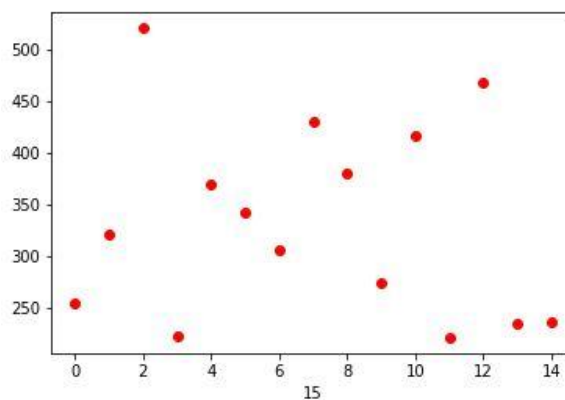
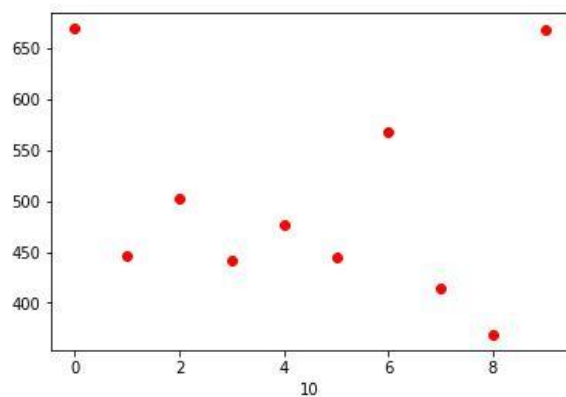
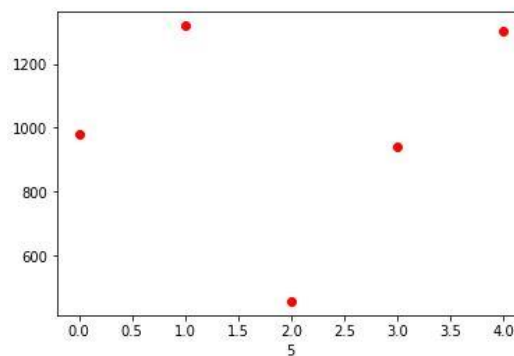
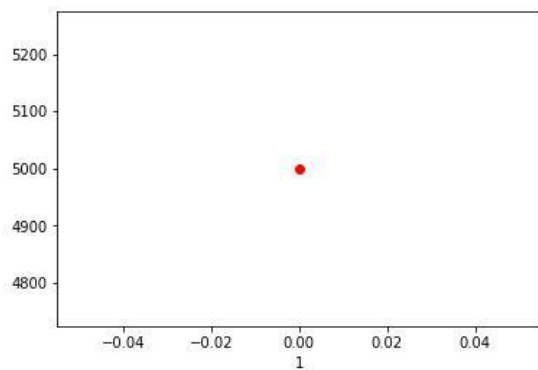


پ) همین منحنی را به ازای $k=16$ میخواند

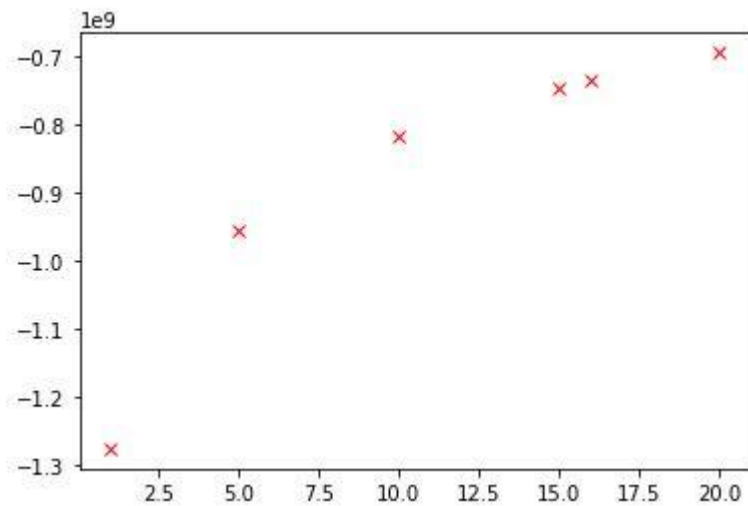


همانطور که مشاهده میشود میزان روند کلی نزولی (کاهش فاصله = افزایش امتیاز score) است و با افزایش میزان کلاستر عدد همگرا شده ، کاهش میابد.

ت) اجرای الگوریتم به ازای $k=1, 5, 10, 15, 16, 20$ و نمونه های موجود در آن



میزان score یا همان قرینه فاصله within distant به ازای $k=1, 5, 10, 15, 16, 20$



و اما در نهایت بررسی دقیق به ازای $k=10$

labeled data	clustering
0 : 507	0 : 471
1 : 570	1 : 659
2 : 530	2 : 267
3 : 514	3 : 711
4 : 479	4 : 247
5 : 418	5 : 478
6 : 462	6 : 515
7 : 530	7 : 625
8 : 517	8 : 612
9 : 472	9 : 414

بیشترین میزان اطلاعات در داده های label خوره متعلق به 1 است پس نتیجه میگیریم که کلاستر 3 متعلق به عدد 1 است

و به همین ترتیب پیش میرویم