باسمه تعالی

تکلیف سری پنجم داده کاوی

سارا برادران (شماره دانشجویی : ۹۶۲۴۱۹۳)

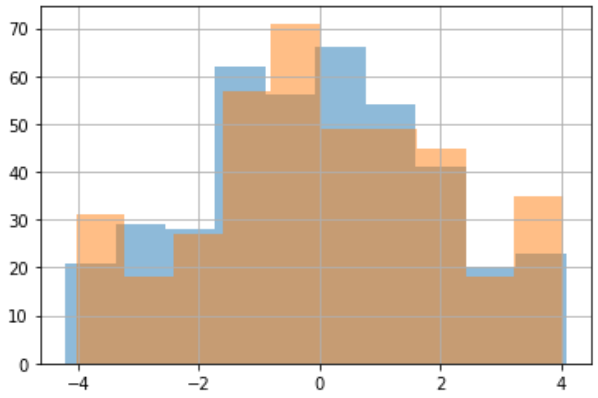
سوال اول (شبکه عصبی)

a) فایل csv را با کمک متد read_csv() کتابخانه pandas به دیتافریم تبدیل می کنیم.

b) در دیتافریم موجود missing value ها بررسی شده و مقادیر ستون ها فاقد missing value هستند. رنج مقادیر هر ستون را مورد بررسی قرار می دهیم و رنج مقادیر به صورت زیر می باشد :

```
X1      ( min =  -4.211898112302497  max =  4.078165875644617 )
X2      ( min =  -4.035124003739587  max =  4.037643470481511 )
Class   ( min =  0  max =  1 )
```

به علاوه توزیع مقادیر متغیرهای X1 و X2 به صورت زیر می باشد که به حالت نرمال نزدیک است.



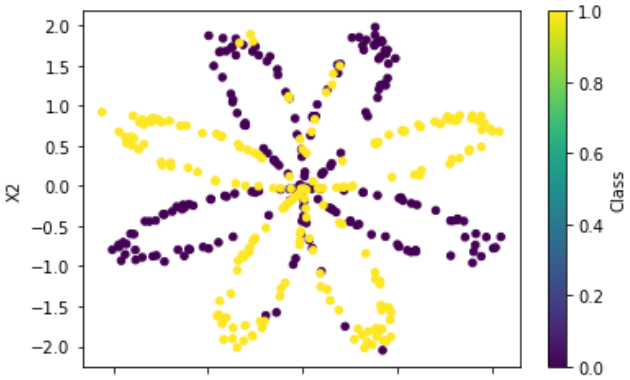
با توجه به مقادیر ذکر شده تنها از متد StandardScaler برای استاندارد سازی داده ها استفاده شده است که این امر رنج متغیر ها را تاحدودی محدود تر می کند اما توزیع مقادیر را تغییری نمی دهد.

```
X1      ( min =  -2.11872062930916  max =  2.094811727652321 )

X2      ( min =  -2.0481666654602075  max =  1.980981188795628 )

Class   ( min =  0  max =  1 )
```

c) نمودار scatter plot رسم شده به صورت زیر می باشد. که در آن نقاط زرد رنگ متعلق به داده هایی است که ستون Class نظیر آن ها 1 بوده و نقاط تیره رنگ متعلق به داده هایی است که ستون Class نظیر آن ها 0 بوده است.



(d

```
x = Flower_std[['X1', 'X2']]
```

```
y = Flower_std[['Class']]
```

e) به کمک متد train_test_split و با تنظیم پارامتر test_size = 0.2 داده های یادگیری و تست را جدا سازی کرده و توزیع مقادیر متغیر Class در y_train و y_test به صورت زیر می باشد.

```
0    163
1    157
Name: Class, dtype: int64
===========> train <===========
Distribution Of 0 : 50.94 %
Distribution Of 1 : 49.06 %


1    43
0    37
Name: Class, dtype: int64
============> test <============
Distribution Of 0 : 46.25 %
Distribution Of 1 : 53.75 %
```

توزیع مقادیر 1 , 0 در داده های تست و یادگیری تقریبا مشابه و به میزان حدودا یکسان است و اختلاف چندانی ندارد به طور تقریبی نیمی از داده های تست و یادگیری دارای مقدار Class=0 و نیم دیگر دارای مقدار Class=1 است.

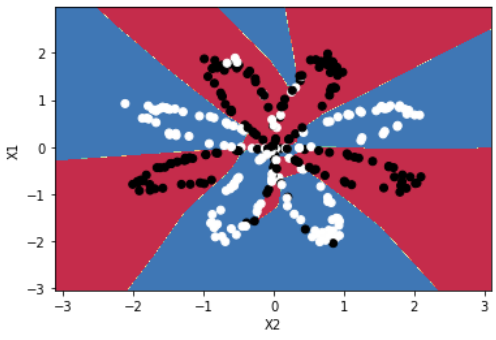
f) پارامترهای MLPClassifier را به صورت زیر تنظیم می کنیم:

```
mlp = MLPClassifier(hidden_layer_sizes=(20, 20, 20), max_iter=1000, verbose=False)
```

در این MLPClassifier تعداد ۳ لایه مخفی در نظر گرفته شده است که هر لایه مخفی ۲۰ نود را شامل می شوند. به علاوه max_iteration برابر ۱۰۰۰ تنظیم شده است که همان تعداد مراحل وزن دهی و تست وزن ها برای همگرا شدن شبکه عصبی به وزن دهی ثابت و مناسب مدل است.

g) دقت مدل توسط متد accuracy_score اندازه گیری شده است که این مقدار 0.8875 می باشد به این معنی که مدل حدود ۸۸ درصد دقت در داده های تست از خود نشان داده است.

h) نمودار رسم شده به صورت زیر می باشد که در آن مرز میان دسته های گوناگون مشخص شده است در این شکل داده هایی که با نقاط سفید رنگ در زمینه آبی رسم شده اند در Class = 1 و داده هایی که با نقاط تیره و در زمینه قرمز رسم شده اند در Class = 0 قرار دارند.



i) در سه مرحله یکبار تاثیر سایزهای لایه مخفی بار دیگر تاثیر max_iter و نهایتا تاثیر activation function را بر مدل شبکه عصبی بررسی میکنیم.

در اولین مرحله دقت مدل شبکه عصبی با تعداد 3 لایه مخفی با تعداد نود 20, 25, 30, 35, 40, 45 و max_iter = 1000 و activation = relu سنجیده شده و نتایج زیر بدست آمده است.

```
accuracy_score for hidden layer  20 = 0.9
accuracy_score for hidden layer  25 = 0.8875
accuracy_score for hidden layer  30 = 0.9125
accuracy_score for hidden layer  35 = 0.8875
accuracy_score for hidden layer  40 = 0.925
accuracy_score for hidden layer  45 = 0.875
```

در دومین مرحله دقت مدل شبکه عصبی با تعداد 3 لایه مخفی و تعداد نود ثابت 20 در هرلایه و max_iter های 1000 , 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900 و activation = relu سنجیده شده و نتایج زیر بدست آمده است.

```
accuracy_score for max_iter 1000 = 0.875
accuracy_score for max_iter 1100 = 0.9
accuracy_score for max_iter 1200 = 0.9
accuracy_score for max_iter 1300 = 0.8875
accuracy_score for max_iter 1400 = 0.9125
accuracy_score for max_iter 1500 = 0.9
accuracy_score for max_iter 1600 = 0.8875
accuracy_score for max_iter 1700 = 0.8875
accuracy_score for max_iter 1800 = 0.8875
accuracy_score for max_iter 1900 = 0.8875
```

در سومین مرحله دقت مدل شبکه عصبی با تعداد 3 لایه مخفی و تعداد نود ثابت 20 در هرلایه و max_iter = 1000 و activation های ['identity', 'logistic', 'tanh', 'relu'] سنجیده شده و نتایج زیر بدست آمده است.

```
accuracy_score for activation function identity = 0.4375
accuracy_score for activation function logistic = 0.4625
accuracy_score for activation function tanh = 0.9
accuracy_score for activation function relu = 0.9
```

عملکرد هر یک از این activation function ها به صورت زیر می باشد :

- 'identity', no-op activation, useful to implement linear bottleneck, returns f(x) = x
- 'logistic', the logistic sigmoid function, returns f(x) = 1 / (1 + exp(-x)).
- 'tanh', the hyperbolic tan function, returns f(x) = tanh(x).
- 'relu', the rectified linear unit function, returns f(x) = max(0, x)

با توجه به نتایج بدست آمده به نظر میرسد تاثیر activation function بیش از دو مورد دیگر است چرا که تنها با تغییر activation از موارد identity , logistic به tanh , relu میزان دقت حدودا دوبرابر شده است.

از میان نتایج فوق بهترین پارامتر های هر مرحله را انتخاب کرده و از ترکیب آن ها مدل زیر را می سازیم که دقت آن حدود ۹۱ درصد بدست آمده است.

```
mlp = MLPClassifier(hidden_layer_sizes=(40, 40, 40), max_iter=1400, activation = 'tanh')
Approximatly the best accuracy_score =  0.9125
```

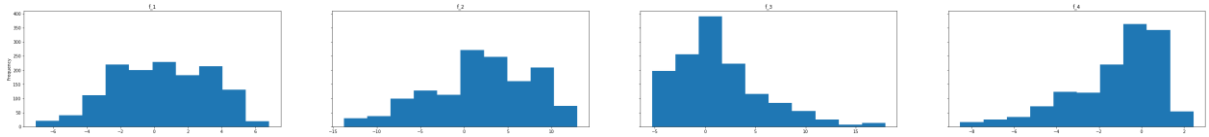
سوال دوم (خوشه بندی)

a) فایل csv را با کمک متد read_csv() کتابخانه pandas به دیتافریم تبدیل می کنیم. و ستون ها را به ترتیب به صورت cols=['f_1', 'f_2', 'f_3', 'f_4', 'Class'] نام گذاری می کنیم.

b) در دیتافریم موجود missing value ها بررسی شده و مقادیر ستون ها فاقد missing value هستند. رنج مقادیر هر ستون را مورد بررسی قرار می دهیم و رنج مقادیر به صورت زیر می باشد :

```
============== The range of columns ==============
f_1       min =  -7.0421
          max =  6.8248
==================================================
f_2       min =  -13.7731
          max =  12.9516
==================================================
f_3       min =  -5.2861
          max =  17.9274
==================================================
f_4       min =  -8.5482
          max =  2.4495
==================================================
Class     min =  0
          max =  1
==================================================
```

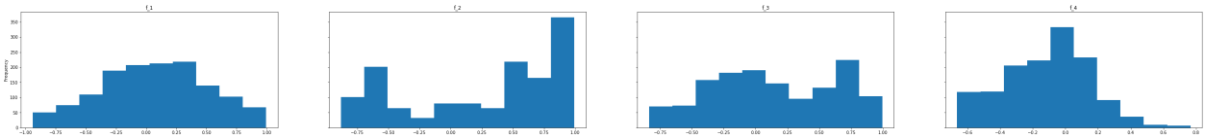
به علاوه توزیع مقادیر متغیرهای f_1 , f_2 , f_3 , f_4 در درون دیتاست Banknote به صورت زیر می باشد



همانطور که از هیستوگرام های بدست آمده مشخص است هیستوگرام نظیر مقادیر ستون f_4 دارای کجی چپ و هیستوگرام نظیر مقادیر ستون f_3 دارای کجی راست است. به علاوه مقادیر بدست آمده در ستون های مختلف دیتاست بسیار متفاوت است لذا نیاز است به گونه ای هم توزیع داده ها را نرمال کرده و هم رنج مقادیر را یکسان نماییم.

به این سبب از متد normalize استفاده کرده و وضعیت مقادیر پس از normalize به صورت زیر خواهد بود.

```
====== The range of columns after normalize ======
f_1       min =  -0.9415817879791171
          max =  0.9952311995179948
==================================================
f_2       min =  -0.882786685144003
          max =  0.9934265193812618
==================================================
f_3       min =  -0.8391798988659801
          max =  0.9975783144233962
==================================================
f_4       min =  -0.6645678813760538
          max =  0.7666352241386792
==================================================
Class     min =  0
          max =  1
==================================================
```



مطابق آنچه در بالا مشخص است کجی هیستوگرام ها برطرف و به حالت نرمال نزدیکتر شده است به علاوه رنج داده های همه ستون ها در بازه 1- تا 1 قرار گرفته است.

c) از متد Kmeans استفاده کرده و پارامتر n_clusters را برابر ۲ قرار می دهیم :

```
Banknote_normalize.drop(columns=['Class'], inplace=True)

labels = kmeans01.fit_predict(Banknote_normalize)
```

d)مقادیر centroid را بدست می آوریم این مقادیر مطابق زیر است :

```
centroids = kmeans01.cluster_centers_
```

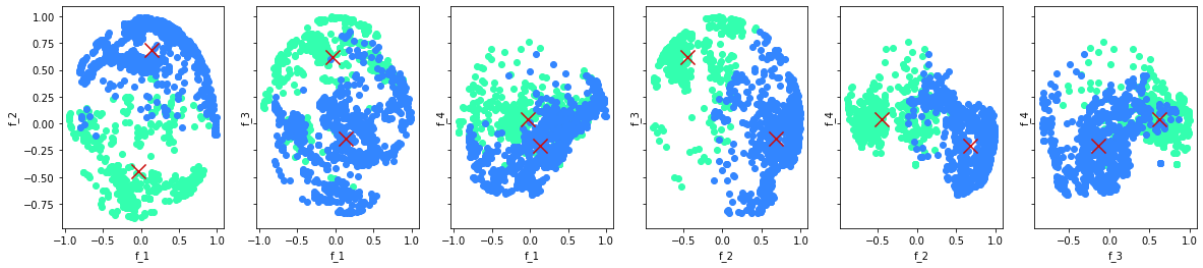
```
[[ 0.09843252  0.70660565 -0.16639498 -0.22916763]
 [ 0.03186233 -0.39858423  0.60673934  0.05887376]]
```

به علاوه توصیف دو دسته ایجاد شده از ظریق متد kmeans به صورت زیر می باشد :

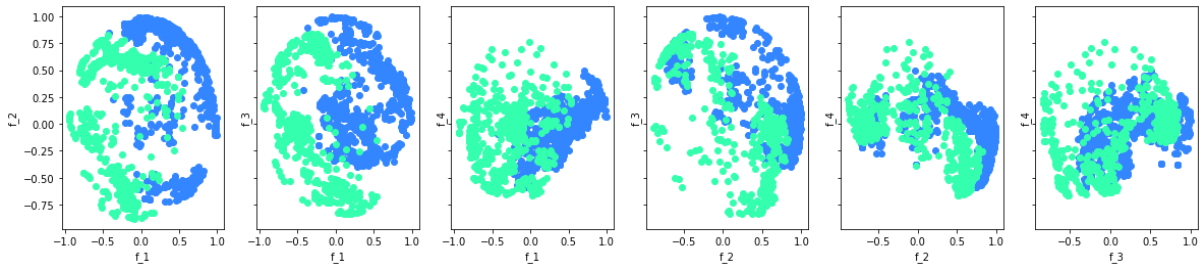
	f_1	f_2	f_3	f_4
count	819.000000	819.000000	819.000000	819.000000
mean	0.098433	0.706606	-0.166395	-0.229168
std	0.416521	0.214910	0.313699	0.247180
min	-0.806218	-0.112175	-0.839180	-0.664568
25%	-0.155978	0.568103	-0.331605	-0.401740
50%	0.147557	0.765298	-0.168925	-0.259896
75%	0.364514	0.883227	0.047908	-0.057960
max	0.995231	0.993427	0.684985	0.649722

	f_1	f_2	f_3	f_4
count	553.000000	553.000000	553.000000	553.000000
mean	0.031862	-0.398584	0.606739	0.058874
std	0.463481	0.339134	0.263088	0.176076
min	-0.941582	-0.882787	-0.587154	-0.380269
25%	-0.236109	-0.663546	0.467560	-0.062539
50%	-0.116652	-0.529924	0.660250	0.035993
75%	0.386021	-0.098141	0.772263	0.151838
max	0.990616	0.580464	0.997578	0.766635

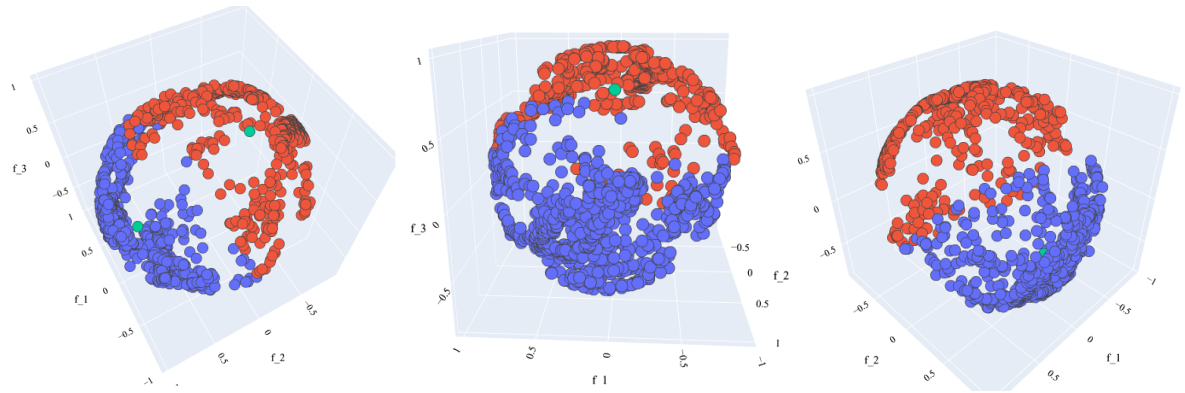
e) Scatter plot نظیر خوشه بندی بر اساس دو به دو ستون ها ایجاد شده و مراکز دسته ها را با ضربدر قرمز زنگ نشان داده شده است :



میتوانیم نمودار های بدست آمده فوق را با Actual Classes مقایسه کنیم. نمودار های زیر دسته بندی داده ها را برحسب متغیر Class موجود در دیتاست نشان می دهد.



نمودار ۳ بعدی برای خوشه های ایجاد شده از طریق kmeans و براساس ستون های f_1 و f_2 و f_3 نیز به صورت زیر می باشد. (مراکز دسته ها با دایره سبز رنگ مشخص شده است.)



f)

algorithm*{"auto", "full", "elkan"}, default="auto"*
K-means algorithm to use. The classical EM-style algorithm is "full". The "elkan" variation is more efficient on data with well-defined clusters, by using the triangle inequality. However it's more memory intensive due to the allocation of an extra array of shape (n_samples, n_clusters).

For now "auto" (kept for backward compatibiliy) chooses "elkan" but it might change in the future for a better heuristic.

g) متد ها و پارامتر های ارزیابی خوشه بندی

Inertia : این متد ارزیابی تعیین می کند نقاط درون یک خوشه تا چه اندازه از هم فاصله دارند. همانطور که میدانیم یکی از اهداف خوشه بندی مناسب به حداقل رساندن فاصله نقاط درون خوشه ای است. مقدار این پارامتر می تواند از ۰ تا بی نهایت افزایش داشته باشد اما طبیعتا خوشه بندی مناسب این مقدار را به حداقل می رساند و لذا خوشه بندی ای مناسب تر است که اینرسی آن به صفر نزدیک تر باشد.

Silhouette : این متد ارزیابی تعیین می کند نقاط درون یک خوشه از نقاط درون خوشه های دیگر چه اندازه فاصله دارد. همانطور که میدانیم یکی دیگر از اهداف خوشه بندی مناسب به حداکثر رساندن فاصله نقاط درون یک خوشه از دیگر خوشه ها است. مقدار این پارامتر می تواند از 1- تا 1 تغییر کند اما طبیعتا خوشه بندی مناسب این مقدار را به حداکثر می رساند و لذا خوشه بندی ای مناسب تر است که Silhouette آن به 1 نزدیک تر باشد.

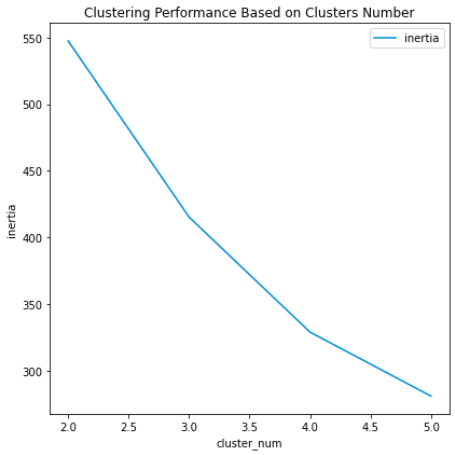
مقدار متد inertia ضمن ارزیابی دسته های بدست آمده برابر زیر می باشد.

```
547.5543725386921
```

h) اگر تعداد کلاس ها را از ۲ تا ۵ افزایش دهیم مقدار inertia بدست آمده برای هر دسته بندی به صورت زیر می باشد.

```
inertia for clusters number between 2 to 5
[547.5543725386921, 415.33928068366987, 329.00583335865804, 281.0030656
347428]
```

i) به نظر میرسد با افزایش تعداد دسته ها مقدار inertia کاهش یافته است. این موضوع از طریق نمودار زیر نیز قابل برداشت است. با توجه به این امر و هم از لحاظ شهودی به نظر میرسد هر چه تعداد دسته های بیشتری داشته باشیم فاصله درون خوشه ای کمتر خواهد شد چرا که هرچه تعداد خوشه ها بیشتر شود تنها داده هایی ک بسیار نزدیک هستند میتوانند در یک خوشه واقع شوند حتی این تعداد خوشه بندی می تواند تا جایی زیاد شود ک هر تک داده در یک خوشه مجزا قرار گیرد که در این حالت هم فاصله درون خوشه ای به صفر خواهد رسید و اینرسی در مطلوب ترین حالت خود واقع می شود. لذا بنابر این پارامتر بهترین دسته بندی در این مثال همان تبدیل داده ها به ۵ خوشه است. به علاوه بیشترین میزان تغییر اینرسی زمانی رخ داده است که تعداد خوشه ها از ۲ به ۳ افزایش داشته است و شیب تغییرات اینرسی ضمن افزایش تعداد خوشه ها در مراحل بعدی کمتر است.



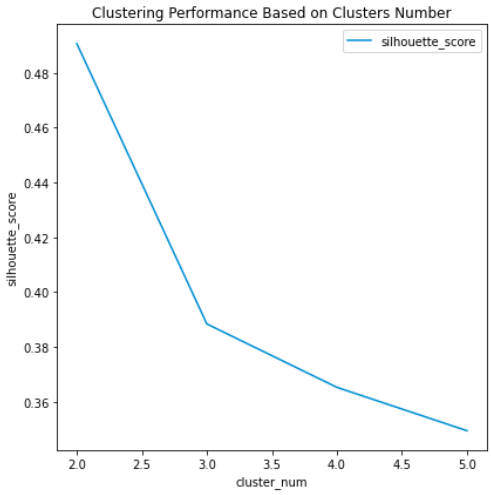
j)مقدار متد silhouette_score ضمن ارزیابی دسته های بدست آمده برابر زیر می باشد.

```
0.49065453916117746
```

اگر تعداد کلاس ها را از ۲ تا ۵ افزایش دهیم مقدار silhouette_score بدست آمده برای هر دسته بندی به صورت زیر می باشد.

```
silhouette for clusters number between 2 to 5
[0.49065453916117746, 0.38837650192531115, 0.3652530151508478, 0.349425
7466430939]
```

به نظر میرسد با افزایش تعداد دسته ها مقدار silhouette_score کاهش یافته است. این موضوع از طریق نمودار زیر نیز قابل برداشت است. اما نقطه ماکسیمم در ۲ خوشه رخ داده است با توجه به این امر به نظر میرسد بهترین دسته بندی بنابر این متد تبدیل داده ها به همان ۲ خوشه باشد. نکته حائز اهمیت این است نمیتوان ادعا کرد هرچه تعداد خوشه ها بیشتر باشد فاصله برون خوشه ای مطلقا افزایش می یابد یا مطلقا کاهش می یابد چراکه اگر داده های بسیار نزدیکی که میتوانستند در یک خوشه قرار داده شوند به دلیل وجود تعداد خوشه زیاد در میان خوشه ها پراکنده شوند میزان فاصله برون خوشه ای طبیعتا کاهش می یابد این موضوع در مثال زیر نیز کاملا واضح است. و چنانچه داده های بسیار پراکنده ای داشته باشیم با افزایش تعداد خوشه ها هر یک ازین دسته داده های پراکنده در یک خوشه قرار گرفته و لذا فاصله برون خوشه ای طبیعتا افزایش می یابد. به علاوه بیشترین میزان تغییر silhouette_score زمانی رخ داده است که تعداد خوشه ها از ۲ به ۳ افزایش داشته است و شیب تغییرات silhouette_score ضمن افزایش تعداد خوشه ها در مراحل بعدی کمتر است.



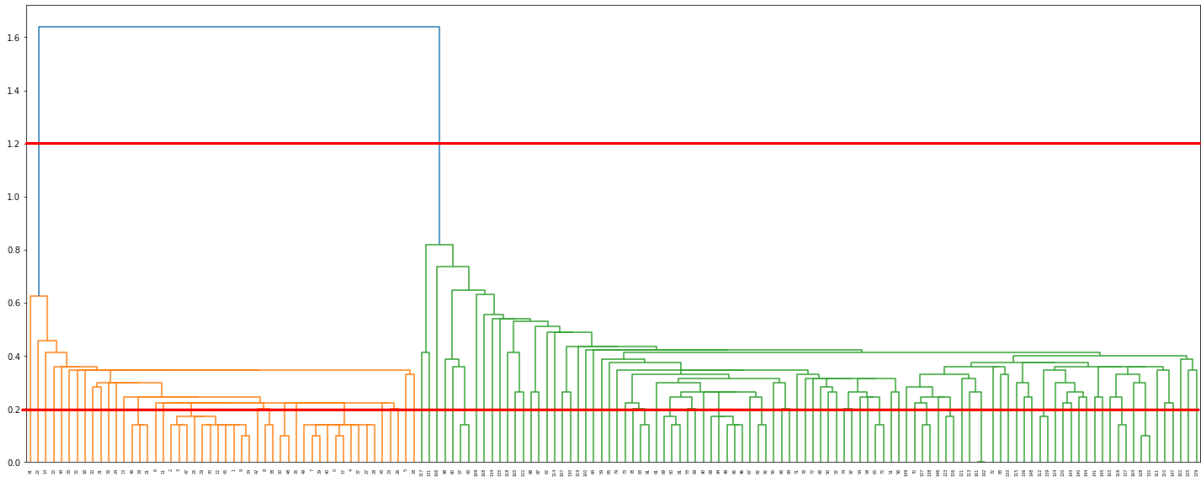
k) این سوال در بخش I و j مفصلا توضیح داده شده است. همانطور که گفته شد بنابر پارامتر inertia بهترین میزان اینرسی در ۵ خوشه بدست آمده است و بنابر متدsilhouette بهترین میزان silhouette در ۲ خوشه بدست آمده است و بنابر توضیحات داده شده لزوما اینرسی کوچک به silhouette بزرگ ختم نمی شود و بالعکس.

سوال سوم (خوشه بندی سلسله مراتبی)

(a

```
iris = load_iris(as_frame=True)
x = iris.data
y = iris.target
Z = linkage(x, 'single')
```

b) همانگونه که در نمودار مشخص است هر چه قطع در سطح بالاتری صورت پذیرفته باشد تعداد خوشه های ایجاد شده کمتر است. و هرچه در سطح پایین تری قطع رخ دهد تعداد خوشه های ایجاد شده به مراتب بیشتر است. اگر خطوط قرمز رسم شده را نگاه کنیم در قطعی که در سطح 1.2 اتفاق افتاده است تنها دو خوشه و در قطعی که در سطح 0.2 اتفاق افتاده است تعداد بسیار زیادی خوشه ایجاد شده است.



criterion='distance'

c)از متد fcluster با پارامتر criterion= distance و level = 6 استفاده شده و نتایج به صورت زیر می باشد:

label = fcluster(Z, 6, criterion='distance')

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1]
```

همانطور که مشخص است تمام داده ها تنها در یک خوشه قرار گرفته اند و هیچگونه تفکیکی صورت نپذیرفته است.

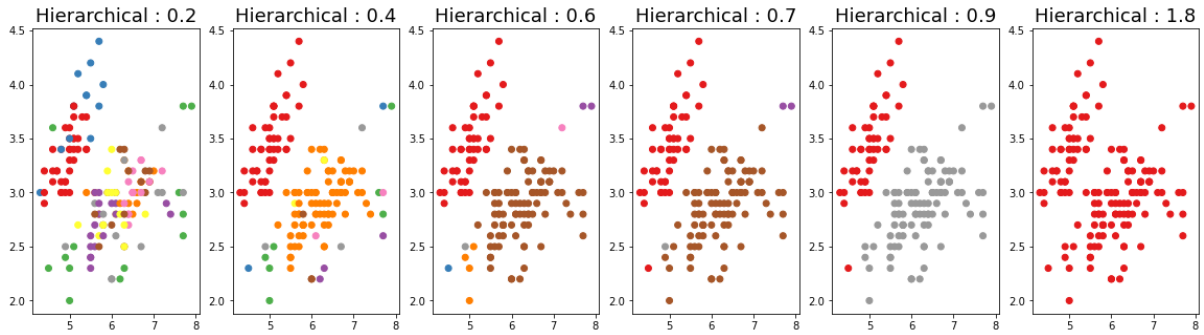
d) حال اگر fcluster را با پارامتر های مشابه و تنها با مقدار level = 0.5 استفاده نماییم نتایج زیر بدست می آید :

```
[ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1
1  1  1  7  7  7  7  7  7  7  4  7  7  4  7  7  7  7  7  7  7  6  7  7
7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  6  7  7  7  7  7  4  7
7  7  7  4  7  7  7  7  7  7  5 12  7 10 11  7  7  7  7  7  7  7  3  5
7  7  7  5  7  7  7  7  7  7  7  7  3  7  7  9  8  7  7  7  7  7  7  7
7  7  7  7  7  7  7]
```

همانطور که مشخص است داده ها در میان ۱۲ خوشه مجزا تقسیم شده اند.

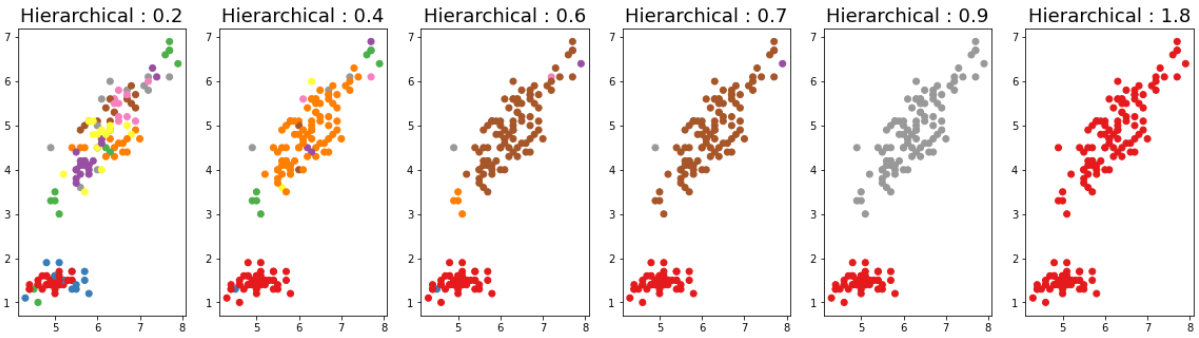
حال به طور دقیق تر تاثیر level بر تعداد خوشه هارا مورد بررسی قرار می دهیم :

در داخل یک حلقه مقدار level را از مقادیر کوچک به بزرگ افزایش داده و نمودار های scatter plot را برحسب ستون های sepal length (cm) و sepal width (cm) رسم کرده ایم که در آن cluster های مختلف با رنگ های گوناگون نمایش داده شده است.

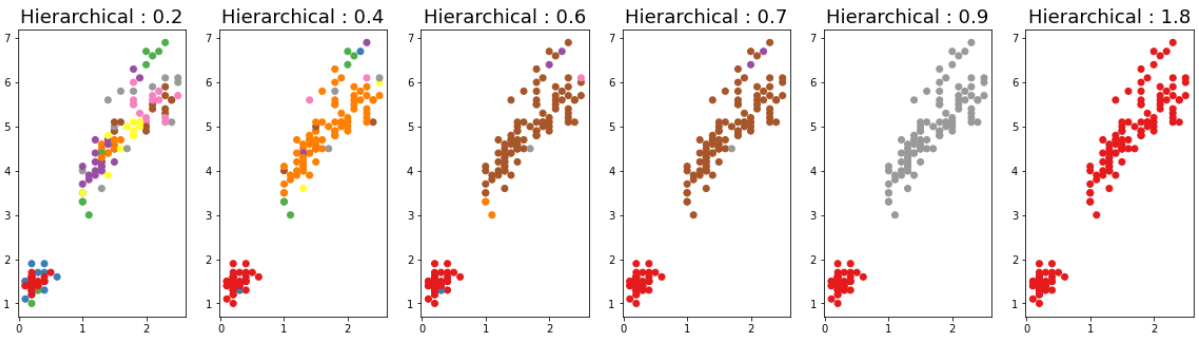


همانگونه که مشخص است در ابتدا زمانی که level = 0.2 بوده تعداد بسیار زیادی خوشه ایجاد شده و داده ها در میان خوشه ها تقسیم شده است در level = 0.4 تعداد خوشه ها نسبت به حالت قبل کاهش یافته در level = 0.6 مجددا نسبت به حالت قبل تعداد خوشه ها کمتر شده و داده ها در میان خوشه های کمتری تقسیم شده به همین ترتیب با افزایش level تعداد خوشه های کمتری ایجاد شده است به طوری که در level = 0.9 تمام داده ها به دو خوشه و در level = 1.8 داده ها تنها به یک خوشه منحصربه فرد تقسیم شده اند. این دقیقا مشابه رفتار نمودار dendrogram است که هرچه قطع در لول بالاتری صورت پذیرد تعداد خوشه های کمتری ایجاد می شود و بالعکس.

همین نمودار برحسب ستون های sepal length (cm) و petal length (cm) رسم شده و نتایج زیر مشابه قسمت بالا است :



همین نمودار برحسب ستون های 'petal width (cm) و petal length (cm) رسم شده و نتایج زیر مشابه قسمت بالا است :



criterion='maxclust'

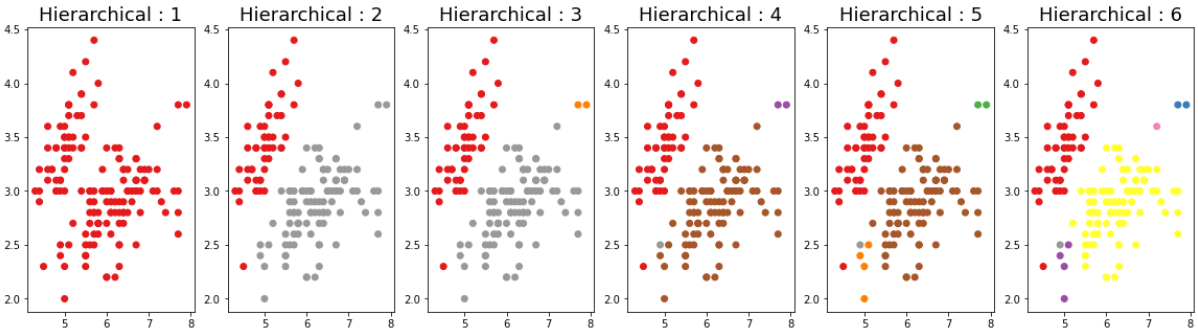
از متد fcluster با پارامتر 'criterion='maxclust و level = 6 استفاده شده و نتایج به صورت زیر می باشد:

همانطور که مشخص است داده ها در میان ۶ خوشه مجزا تقسیم شده اند.

```
label = fcluster(Z, 6, criterion='maxclust')
```

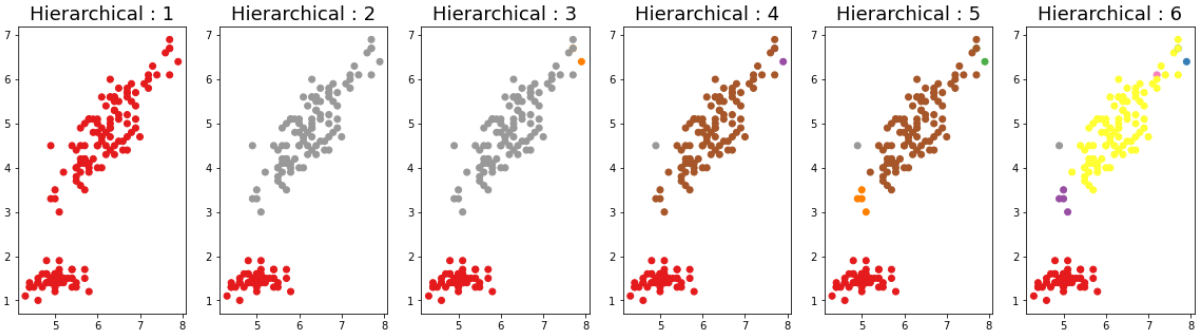
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 4 4 4 4 4 3 4 4 3 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 3 4 4 4 4 4 4 6
4 4 5 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4]
```

d) در داخل یک حلقه مقدار level را از 1 تا 6 افزایش داده و نمودار های scatter plot را برحسب ستون های sepal length (cm) و sepal width (cm) رسم کرده ایم که در آن cluster های مختلف با رنگ های گوناگون نمایش داده شده است.

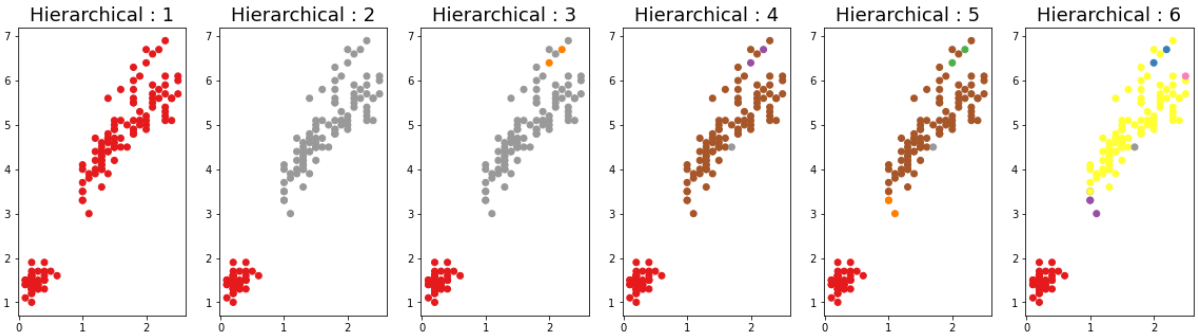


همانگونه که مشخص است در ابتدا زمانی که level = 1 بوده تمام داده ها در یک دسته قرار گرفته اند و هیچگونه تفکیکی صورت نپذیرفته است. در level = 2 داده ها به دو دسته تقسیم شده اند و در level = 3 به گونه ای دقیق تر چند مورد داده ای که فاصله چشمگیری هم از دسته اول و هم از دسته دوم داده های قسمت قبل داشته اند در یک دسته سوم و جداگانه قرار داده شده اند به همین ترتیب هرچه level افزایش یافته است دسته بندی نیز دقیق تر شده است و تعداد دسته ها نیز افزایش یافته است.

همین نمودار برحسب ستون های sepal length (cm) و petal length (cm) رسم شده و نتایج زیر مشابه قسمت بالا است :



همین نمودار برحسب ستون های 'petal width (cm) و petal length (cm) رسم شده و نتایج زیر مشابه قسمت بالا است :



تاثیر افزایش level بر روی تعداد خوشه های ایجاد شده ، در fcluster با پارامتر criterion=maxclust و پارامتر criterion= distanc عکس یکدیگر است.