گزارش پروژه خوشه بندی - درس یادگیری ماشین

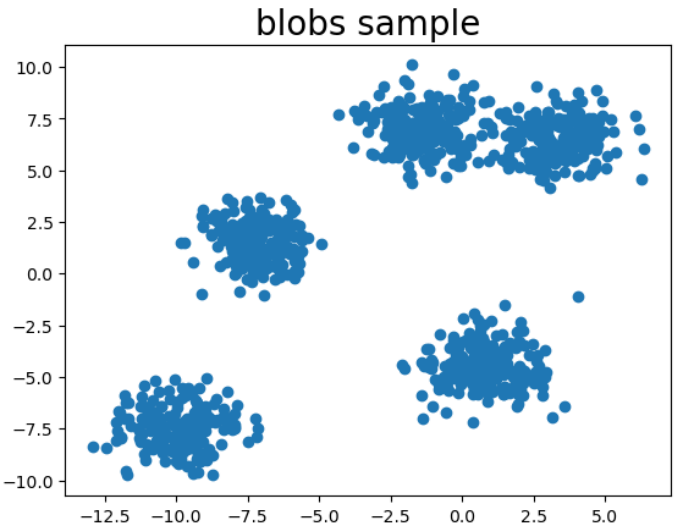
مهدیس رحمانی

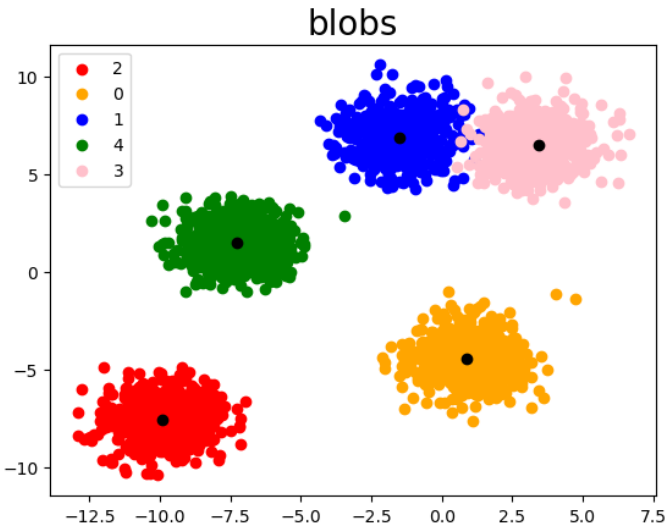
ابتدا 3 نوع داده تولید میکنیم:

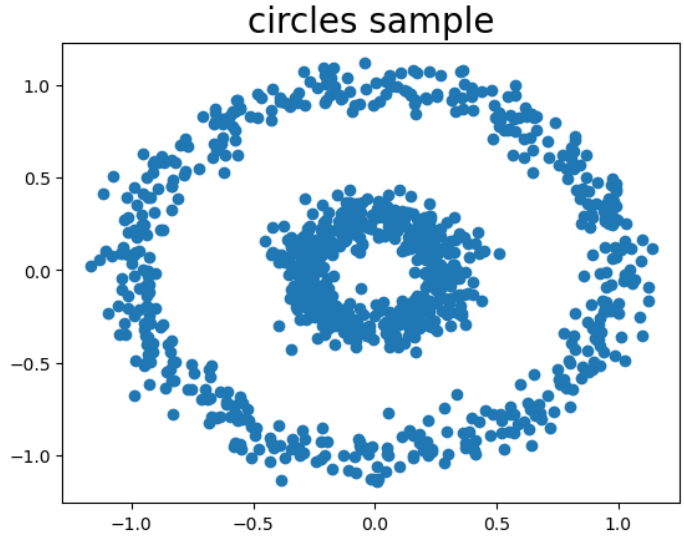
1 – Blobs
2 – Circles
3 – Moons

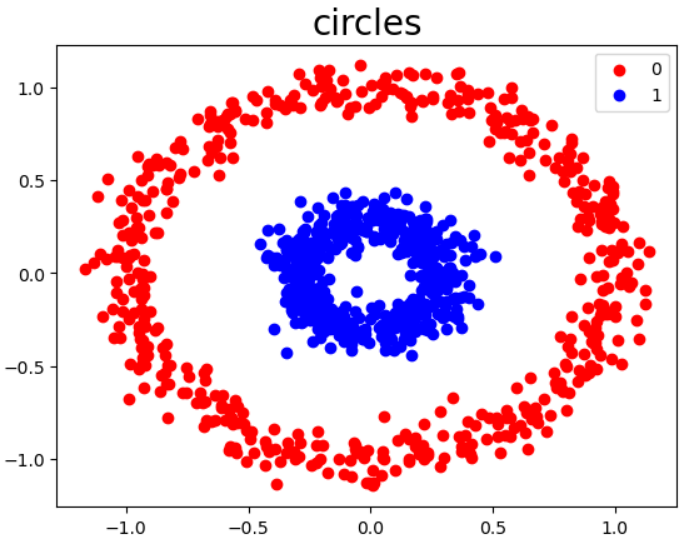
این داده ها با استفاده از کتابخانه سایکیت-لرن تولید شده و باید تعداد سمپل ها، تعداد خوشه ها و استیت رندوم بودن را برای آنها تعیین کنیم.

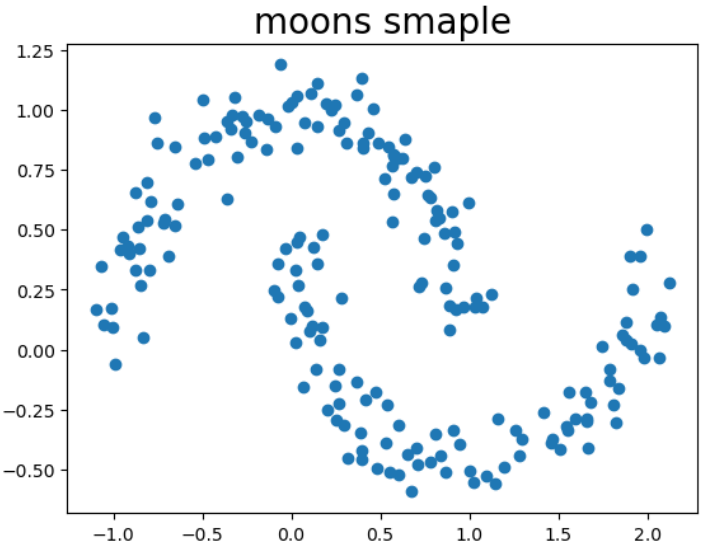
در صفحه ی بعد ابتدا حاصل دیتا های تولید شده بدون رنگ و با رنگ منتسب به هر کلاس آورده شده،
مرکز خوشه ها با رنگ مشکی مشخص شده و در ادامه کد مربوط نیز آورده شده است.

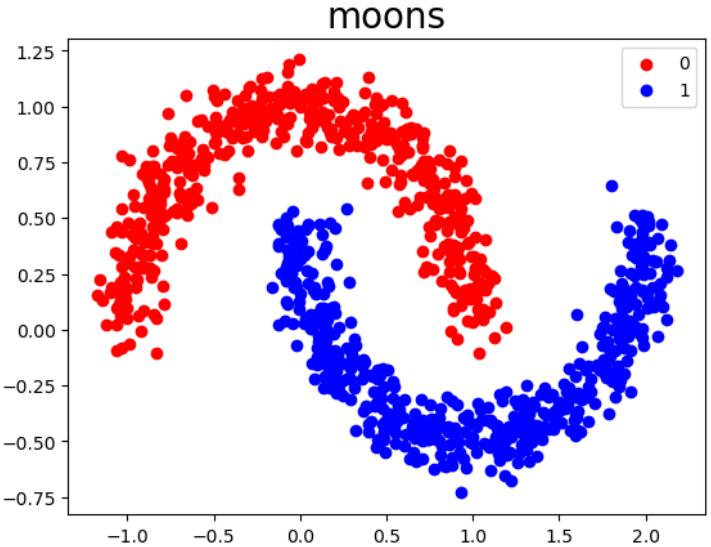

blobs sample


blobs
2
0
1
4
3


circles sample


circles
0
1


moons smaple


moons
0
1

Importing libraries

In [592]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
import pandas as pd
```

sample generators (for testing algorithms)

In [593]:
```
# make blobs
# make moons
# make circles
```

Make blobs

In [594]:
```
n_classes = 5 # n = 5 labels

data, labels, centers = make_blobs(n_samples = 3000,
                                   centers = n_classes,
                                   random_state = 100,
                                   return_centers = True)

# n_features = 2 as default for 2D plotting
```

Make circles

In [319]:
```
from sklearn.datasets import make_circles
```

In [320]:
```
n_circles = 2
c_data, c_labels = make_circles(n_samples = 1000,
                                random_state = 100,
                                noise = 0.07,
                               factor = 0.3)
```

Make moons

In [330]:
```
from sklearn.datasets import make_moons
```

In [331]:
```
m_data, m_labels = make_moons(n_samples = 1000,
                              random_state = 100,
                             noise = 0.09)
```

در ادامه 3 الگوریتم:

1 – K-Means

2 – Agglomerative clustering

3 – DBSCAN

را روی دیتاست های تولید شده اعمال میکنیم و لیبل های حاصل ازین الگوریتم ها را به دیتا فریم های ساخته شده برای دیتاست ها اضافه میکنیم،

اینکار برای ترسیم و مقایسه ی نتایج کار را بسیار راحت تر میکند.

سپس نتایج را برای هر کلاس (خوشه/لیبل) رنگ میکنیم.

4 معیار:

1 – Rand index

2 – Jaccard index

3 – Silhouette index

4 – Davies-Bouldin index

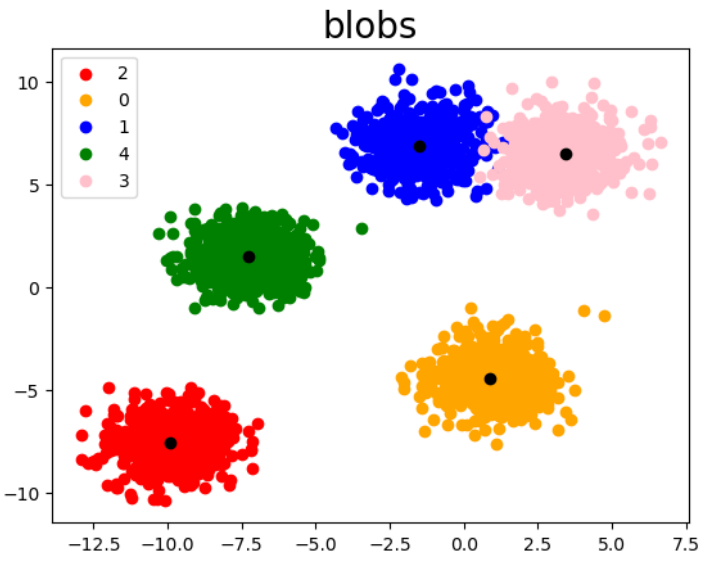
را محاسبه میکنیم.

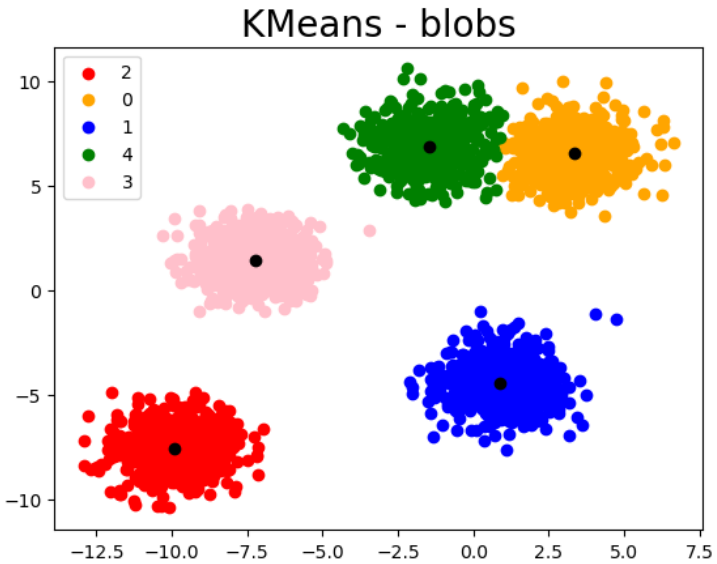
یک نکته ی جالب!

همانطور که در کلاس اشاره شد اگر بخواهیم مثل کلسیفیکیشن از اسکور های مثل اکیورسی استفاده کنیم ممکن است در اسم و لیبل گذاری دچار مشکل شده باشیم.

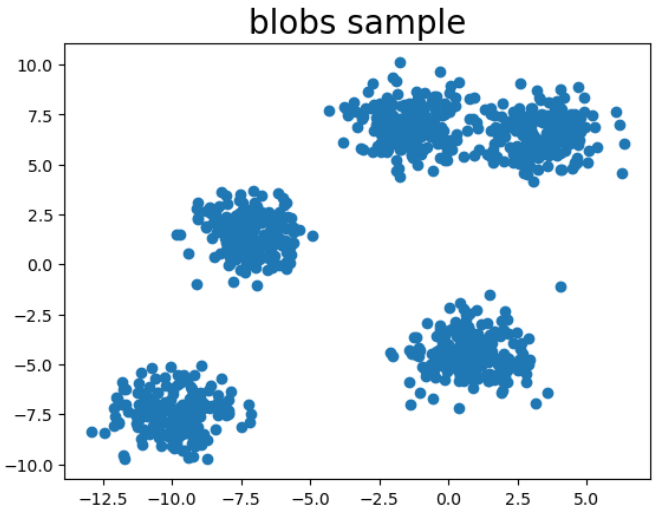
مثلا ما به یک خوشه لیبل صفر داده ایم (در گلدن – استاندارد)

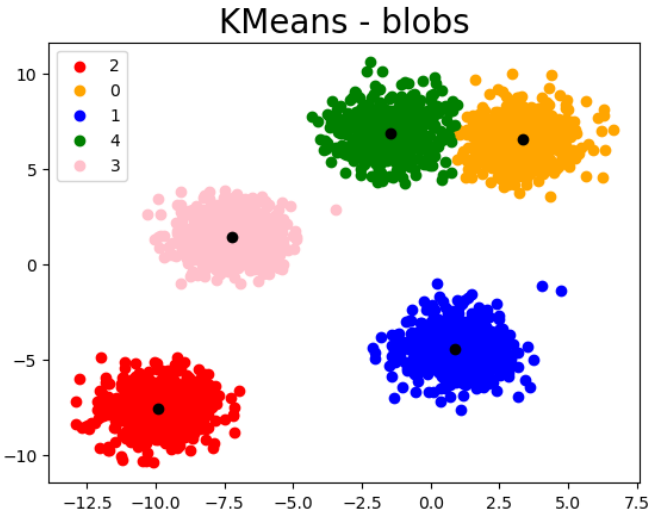
اما سایکیت لرن به آن 1 میدهد:

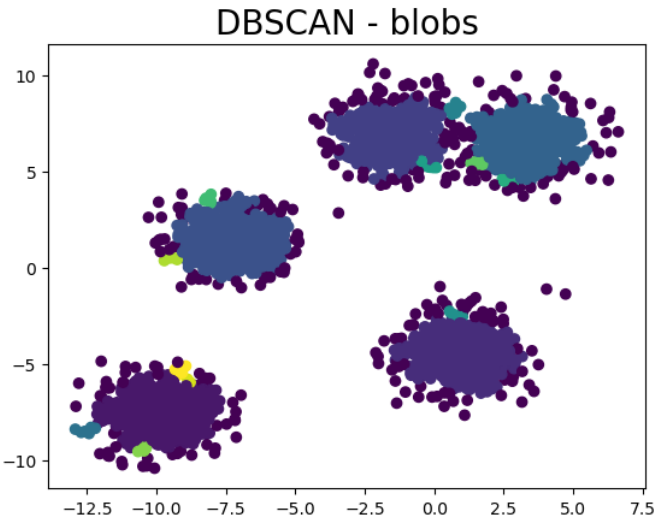


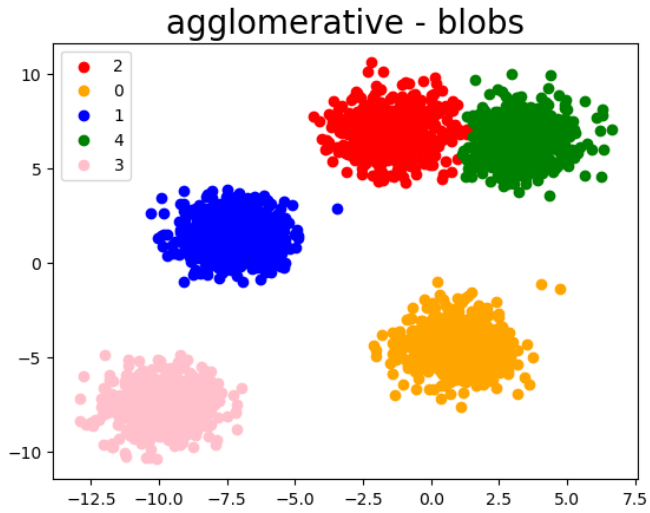


در اینجا تنها خوشه ی قرمز لیبلش حفظ شده است.



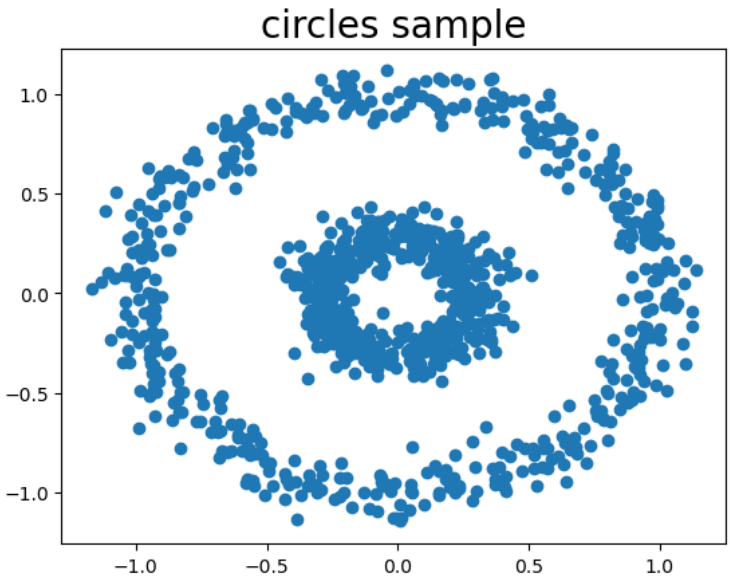


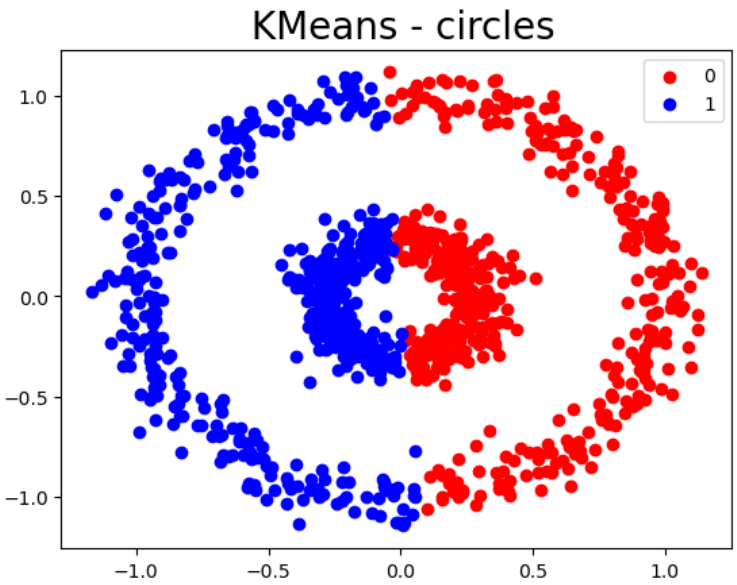


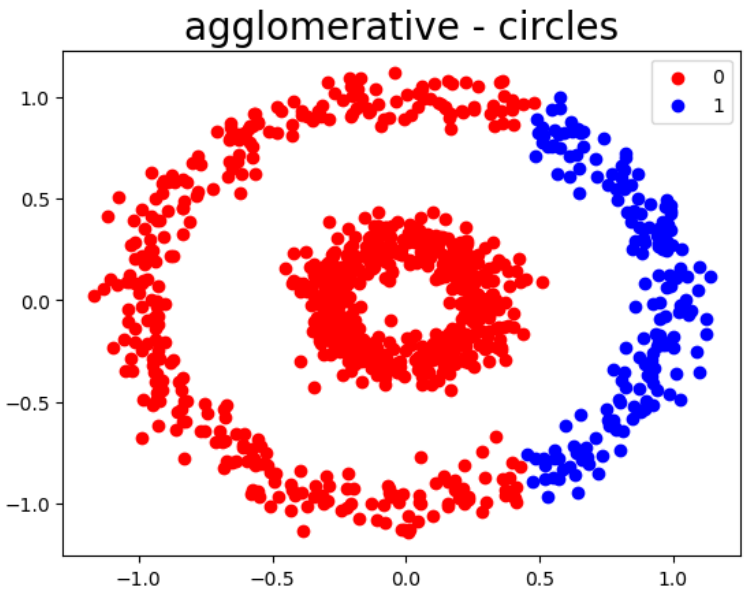


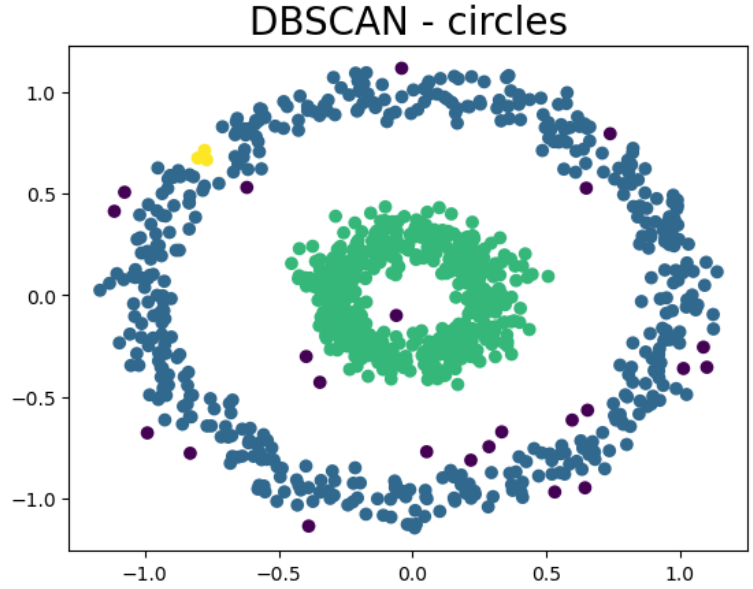
همانطور که در کلاس نیز اشاره شد بهترین مزیت روش دی-بی-اسکن این است که به زور همه ی نقاط را در یک کلاس قرار نداده و نقاط پرت را جدا میسازد و حتی میتواند جدا کلاس بندی کند.

بین دو خوشه ی بالا سمت راست، در روش اگلومریتیو و کی-مینز اختلافاتی هست.

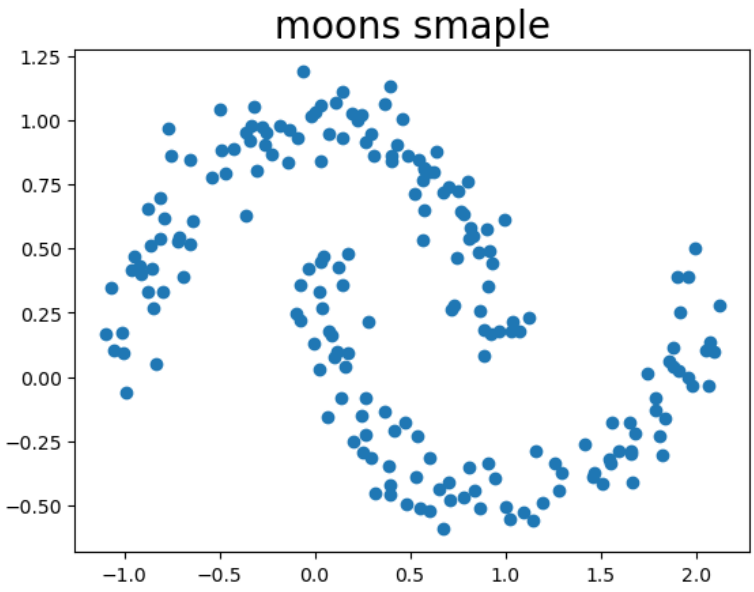


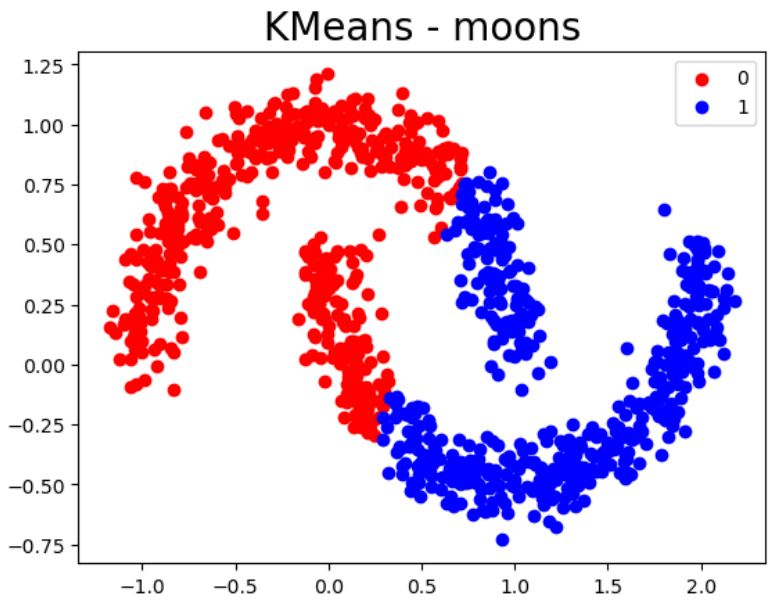


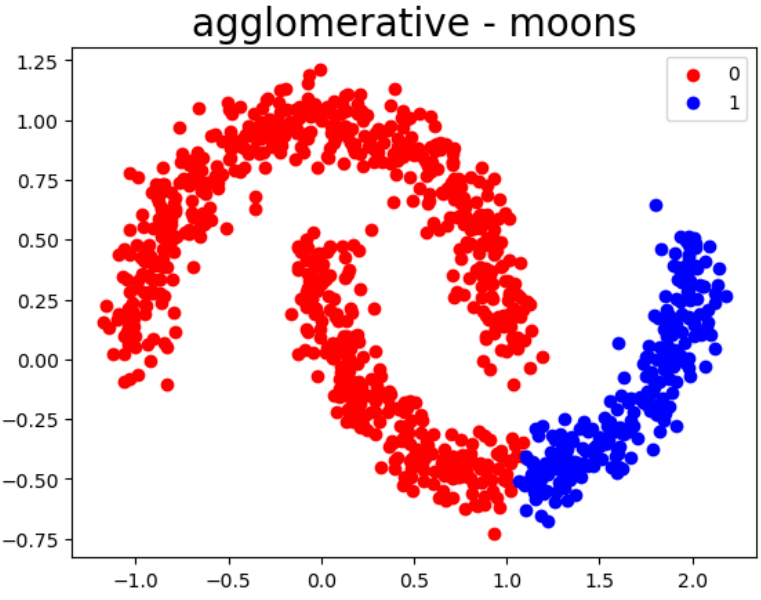


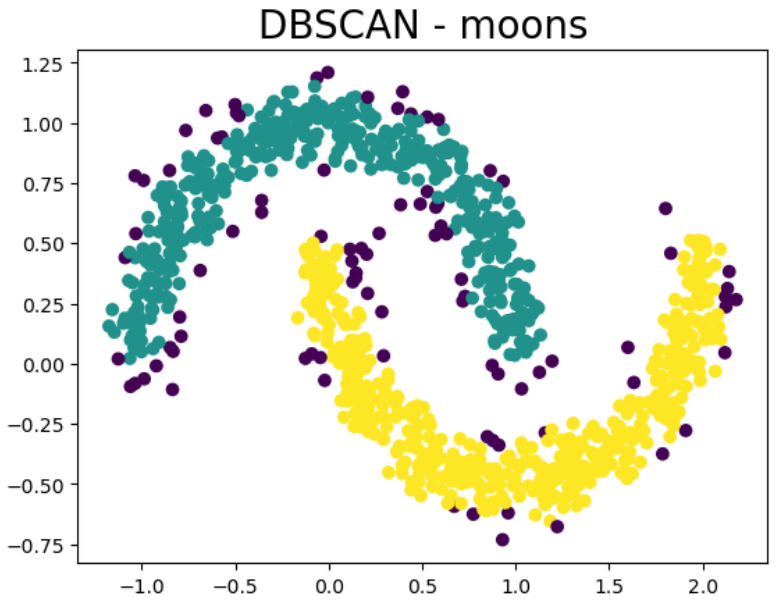


در این دیتاست نیز الگوریتم دی-بی-اسکن از همه موفق تر عمل کرده است، هرچند که این الگوریتم بسیار به شعاع همسایگی و تعداد همسایه ها بسیار حساس است.









در این دیتاست الگوریتم دی-بی-اسکن از همه عملکرد موفق تری داشته است.

محاسبه ی شاخص های عملکرد برای مدل **کی-مینز** :

Metrics

In [576]:
```
from sklearn import metrics
```

blobs

In [577]:
```
rand_index = metrics.adjusted_rand_score(labels, kmeans_model_blobs.labels_)
print("RandIndex for K-means clustering:", rand_index)
```

```
RandIndex for K-means clustering: 0.994192951553166
```

In [586]:
```
jaccard_index = metrics.jaccard_score(labels, kmeans_model_blobs.labels_, average = 'micro')
print("JaccardIndex for K-means clustering:", jaccard_index)
```

```
JaccardIndex for K-means clustering: 0.1111111111111111
```

In [580]:
```
silhouette_index = metrics.silhouette_score(data, kmeans_model_blobs.labels_)
print("SilhouetteIndex for K-means clustering:", silhouette_index)
```

```
SilhouetteIndex for K-means clustering: 0.7311884379035529
```

In [587]:
```
davies_bouldin_index = metrics.davies_bouldin_score(data, kmeans_model_blobs.labels_)
print("Davies-bouldin_index for K-means clustering:", davies_bouldin_index)
```

```
SilhouetteIndex for K-means clustering: 0.3776995379438519
```

circles

In [589]:
```
rand_index = metrics.adjusted_rand_score(c_labels, kmeans_model_circles.labels_)
print("RandIndex for K-means clustering:", rand_index)

jaccard_index = metrics.jaccard_score(c_labels, kmeans_model_circles.labels_, average = 'micro')
print("JaccardIndex for K-means clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(c_data, kmeans_model_circles.labels_)
print("SilhouetteIndex for K-means clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(c_data, kmeans_model_circles.labels_)
print("Davies-bouldin_index for K-means clustering:", davies_bouldin_index)
```

```
RandIndex for K-means clustering: -0.00090186770293864
JaccardIndex for K-means clustering: 0.3289036544850498
SilhouetteIndex for K-means clustering: 0.29545714717761107
Davies-bouldin_index for K-means clustering: 1.3039499475719443
```

ادامه ی محاسبه ی شاخص ها برای مدل کی-مینز برای ماه ها:

moons

In [590]:
```
rand_index = metrics.adjusted_rand_score(m_labels, kmeans_model_moons.labels_)
print("RandIndex for K-means clustering:", rand_index)

jaccard_index = metrics.jaccard_score(m_labels, kmeans_model_moons.labels_, average = 'micro')
print("JaccardIndex for K-means clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(m_data, kmeans_model_moons.labels_)
print("SilhouetteIndex for K-means clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(m_data, kmeans_model_moons.labels_)
print("Davies-bouldin_index for K-means clustering:", davies_bouldin_index)
```

```
RandIndex for K-means clustering: 0.23348872144288577
JaccardIndex for K-means clustering: 0.589825119236884
SilhouetteIndex for K-means clustering: 0.488719055349983
Davies-bouldin_index for K-means clustering: 0.7768798820189866
```

همانطور که انتظار می رود از آنجایی که کی-مینز تنها روی داده های کروی به خوبی عمل میکند، معیار برای بلابز بهتر هستند.

محاسبه ی شاخص ها برای اگلومریتیو:

agglomerative

moons

In [118]:
```
rand_index = metrics.adjusted_rand_score(m_labels, moons_pred_agg)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(m_labels, moons_pred_agg, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(m_data, moons_pred_agg)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(m_data, moons_pred_agg)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.23369316412113603
JaccardIndex for agglomerative clustering: 0.589825119236884
SilhouetteIndex for agglomerative clustering: 0.41971339104118344
Davies-bouldin_index for agglomerative clustering: 0.7155283152882757
```

circles

In [119]:
```
rand_index = metrics.adjusted_rand_score(c_labels, circles_pred_agg)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(c_labels, circles_pred_agg, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(c_data, circles_pred_agg)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(c_data, circles_pred_agg)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.12339497590785209
JaccardIndex for agglomerative clustering: 0.19331742243436753
SilhouetteIndex for agglomerative clustering: 0.3458784988832655
Davies-bouldin_index for agglomerative clustering: 1.1203230047674444
```

blobs

In [120]:
```
rand_index = metrics.adjusted_rand_score(labels, blobs_pred_agg)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(labels, blobs_pred_agg, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(data, blobs_pred_agg)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(data, blobs_pred_agg)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.9933690330934359
JaccardIndex for agglomerative clustering: 0.1111111111111111
SilhouetteIndex for agglomerative clustering: 0.7307713363912088
Davies-bouldin_index for agglomerative clustering: 0.3781733949866922
```

همانطور که در شکل ها قابل مشاهده است، این الگوریتم تنها برای بلابز به خوبی عمل کرد، پس شاخص های بلایز از بقیه ی دیتاست ها بالاتر هستند.

شاخص ها برای الگوریتم دی-بی-اسکن:

circles

In [113]:
```
rand_index = metrics.adjusted_rand_score(c_labels, circles_pred)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(c_labels, circles_pred, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(c_data, circles_pred)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(c_data, circles_pred)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.9496190547200798
JaccardIndex for agglomerative clustering: 0.949317738791423
SilhouetteIndex for agglomerative clustering: 0.1083766585448271
Davies-bouldin_index for agglomerative clustering: 78.13008051591422
```

moons

In [111]:
```
rand_index = metrics.adjusted_rand_score(m_labels, moons_pred)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(m_labels, moons_pred, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(m_data, moons_pred)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(m_data, moons_pred)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.8286698394333877
JaccardIndex for agglomerative clustering: 0.8348623853211009
SilhouetteIndex for agglomerative clustering: 0.18383651737540163
Davies-bouldin_index for agglomerative clustering: 3.1568561212494157
```

blobs

In [114]:
```
rand_index = metrics.adjusted_rand_score(labels, blobs_pred)
print("RandIndex for agglomerative clustering clustering:", rand_index)

jaccard_index = metrics.jaccard_score(labels, blobs_pred, average = 'micro')
print("JaccardIndex for agglomerative clustering:", jaccard_index)

silhouette_index = metrics.silhouette_score(data, blobs_pred)
print("SilhouetteIndex for agglomerative clustering:", silhouette_index)

davies_bouldin_index = metrics.davies_bouldin_score(data, blobs_pred)
print("Davies-bouldin_index for agglomerative clustering:", davies_bouldin_index)
```

```
RandIndex for agglomerative clustering clustering: 0.830255501440546
JaccardIndex for agglomerative clustering: 0.0
SilhouetteIndex for agglomerative clustering: 0.06705331596345686
Davies-bouldin_index for agglomerative clustering: 1.1886471407608032
```

همانطور که دیده شد، عملکرد الگوریتم دی-بی-اسکن برای هر سه دیتاست به شدت خوب بود، درنتیجه شتخص های هر سه دیتاست بسیار بالا هستند که عملکرد این الگوریتم را توصیف میکنند.

پایان