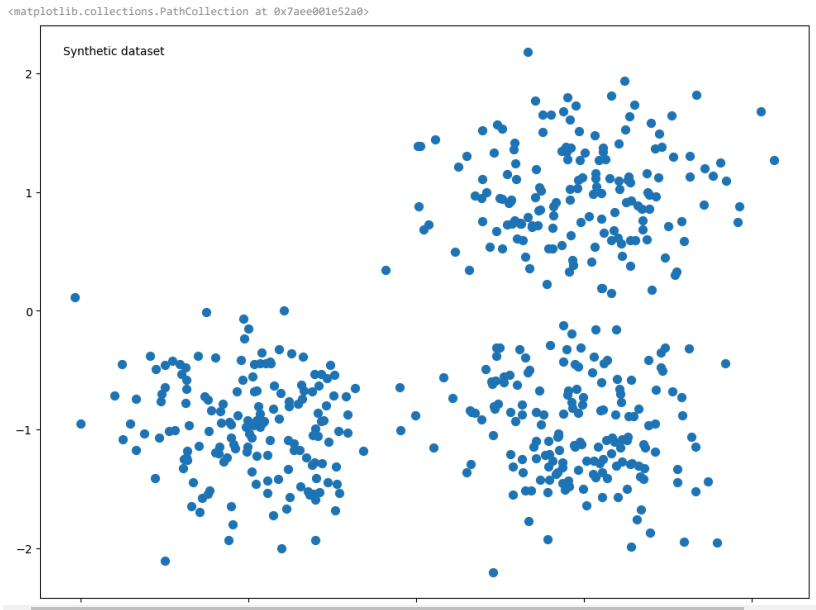
Data creation

First we create the synthetic dataset using make_blobs, same ways as with BIRCH.

```
from sklearn.cluster import DBSCAN, OPTICS
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification, make_moons, make_circles
from matplotlib import pyplot
from sklearn.cluster import KMeans, AgglomerativeClustering, Birch
from sklearn.metrics import silhouette_samples, silhouette_score, calinski_harabasz_score, davies_bouldin_score

from numpy import where
import numpy as np
import matplotlib.pyplot as plt

centers = [[1, 1], [-1, -1], [1, -1]]
X, y_true = make_blobs(n_samples=500, centers=centers, cluster_std=0.4, random_state=31337)

plt.figure(figsize=(12,9))
plt.annotate('Synthetic dataset', xy=(0.03, 0.95), xycoords='axes fraction')
plt.scatter(X[:, 0], X[:, 1], s=50)
```

```
<matplotlib.collections.PathCollection at 0x7aee001e52a0>
```



Clustering

We create the clusters, note that I'm using arbitrary values for both epsilon (eps) and MinPts (min_samples).

```
scaler = StandardScaler()
X = scaler.fit_transform(X)

clusterer = DBSCAN(eps=0.3, min_samples=10, metric='euclidean')
y_pred = clusterer.fit_predict(X)
```

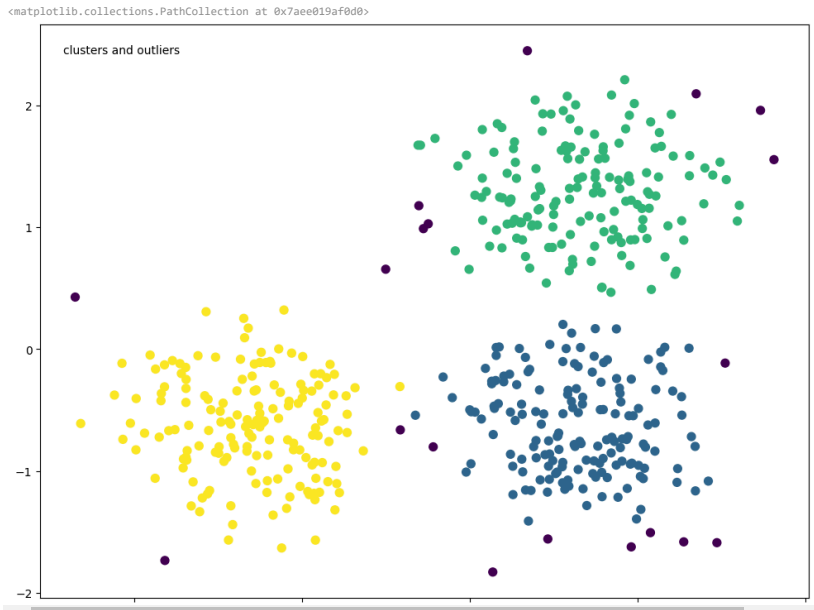
Visualization

Now let's use matplotlib to visualize the identified clusters.

```
plt.figure(figsize=(12,9))
plt.annotate('clusters and outliers', xy=(0.03, 0.95), xycoords='axes fraction')
plt.scatter(X[:, 0], X[:, 1], c=y_pred, s=50)
```



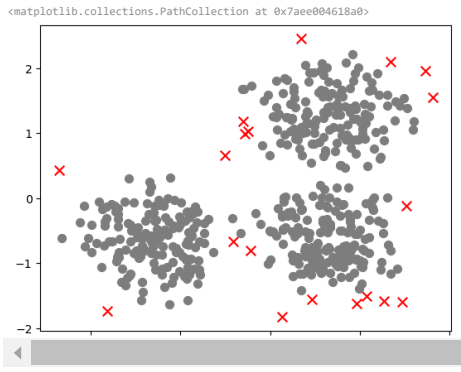
```
<matplotlib.collections.PathCollection at 0x7aee019af0d0>
```



You can see that the outliers are not included in the clusters, we can visualize them specifically:

```
plt.scatter(X[y_pred==-1, 0], X[y_pred==-1, 1], c='red', s=70, marker='x')
plt.scatter(X[y_pred!=-1, 0], X[y_pred!=-1, 1], c='gray', s=50)
```

```
<matplotlib.collections.PathCollection at 0x7aee004618a0>
```



Exercise

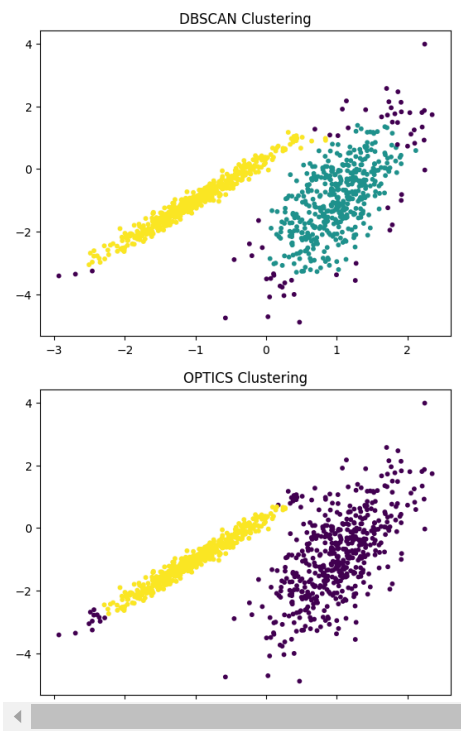
- Apply DBSCAN and OPTICS to the same dataset as BIRCH. Visualize the results. Visualize the outliers.
- Create a synthetic datset with the datasets.make_circles method in sklearn. Apply K-means clustering, BIRCH, DBSCAN and OPTICS to the dataset.
- Create a synthetic datset with the datasets.make_moons method in sklearn. Apply K-means clustering, BIRCH, DBSCAN and OPTICS to the dataset.
- Evaluate the results with the metrics seen the previous day. Which metrics can you use in these datasets? Which algorithm is working better?

1. Apply DBSCAN and OPTICS to the same dataset as BIRCH. Visualize the results. Visualize the outliers.

```
X, y = make_classification(n_samples=1000, n_features=2, n_informative=2, n_redundant=0, n_clusters_per_class=1, random_state=4)
# for class_value in range(2):
#   # get row indexes for samples with this class
#   row_ix = where(y == class_value)
#   # create scatter of these samples
#   pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# # show the plot
# pyplot.show()
```

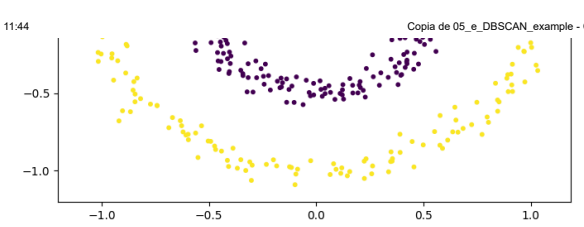


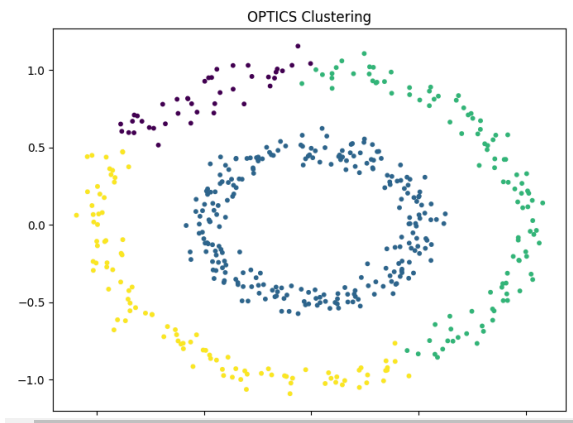
```
dbscan = DBSCAN(eps=0.3, min_samples=10).fit(X)
dbscan_labels = dbscan.labels_

optics = OPTICS(min_samples=10, min_cluster_size=0.45).fit(X)
optics_labels = optics.labels_

pyplot.scatter(X[:, 0], X[:, 1], c=dbscan_labels, cmap='viridis', s=10)
pyplot.title("DBSCAN Clustering")
pyplot.show()

pyplot.scatter(X[:, 0], X[:, 1], c=optics_labels, cmap='viridis', s=10)
pyplot.title("OPTICS Clustering")
pyplot.show()
```



2. Create a synthetic datset with the datasets.make_circles method in sklearn. Apply K-means clustering, BIRCH, DBSCAN and OPTICS to the dataset.

```
X, y = make_circles(n_samples=500, noise=0.05, factor=0.5, random_state=42)

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', s=10)
plt.title("Original Dataset")
plt.show()

kmeans = KMeans(n_clusters=2, random_state=42).fit(X)
kmeans_labels = kmeans.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis', s=10)
plt.title("K-Means Clustering")
plt.show()

birch = Birch(n_clusters=2).fit(X)
birch_labels = birch.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=birch_labels, cmap='viridis', s=10)
plt.title("BIRCH Clustering")
plt.show()

dbscan = DBSCAN(eps=0.2, min_samples=5).fit(X)
dbscan_labels = dbscan.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=dbscan_labels, cmap='viridis', s=10)
plt.title("DBSCAN Clustering")
plt.show()

optics = OPTICS(min_samples=5, xi=0.05, min_cluster_size=0.1).fit(X)
optics_labels = optics.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=optics_labels, cmap='viridis', s=10)
plt.title("OPTICS Clustering")
plt.show()
```



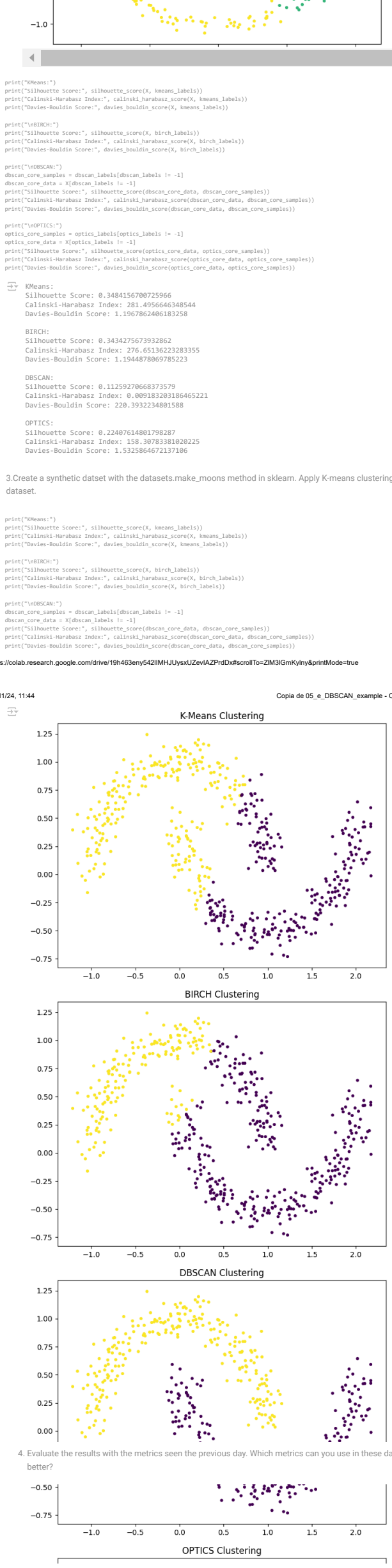



```
print("KMeans:")
print("Silhouette Score:", silhouette_score(X, kmeans_labels))
print("Calinski-Harabasz Index:", calinski_harabasz_score(X, kmeans_labels))
print("Davies-Bouldin Score:", davies_bouldin_score(X, kmeans_labels))

print("\nBIRCH:")
print("Silhouette Score:", silhouette_score(X, birch_labels))
print("Calinski-Harabasz Index:", calinski_harabasz_score(X, birch_labels))
print("Davies-Bouldin Score:", davies_bouldin_score(X, birch_labels))

print("\nDBSCAN:")
dbscan_core_samples = dbscan_labels[dbscan_labels != -1]
dbscan_core_data = X[dbscan_labels != -1]
print("Silhouette Score:", silhouette_score(dbscan_core_data, dbscan_core_samples))
print("Calinski-Harabasz Index:", calinski_harabasz_score(dbscan_core_data, dbscan_core_samples))
print("Davies-Bouldin Score:", davies_bouldin_score(dbscan_core_data, dbscan_core_samples))

print("\nOPTICS:")
optics_core_samples = optics_labels[optics_labels != -1]
optics_core_data = X[optics_labels != -1]
print("Silhouette Score:", silhouette_score(optics_core_data, optics_core_samples))
print("Calinski-Harabasz Index:", calinski_harabasz_score(optics_core_data, optics_core_samples))
print("Davies-Bouldin Score:", davies_bouldin_score(optics_core_data, optics_core_samples))
```

```
KMeans:
Silhouette Score: 0.3484156700725966
Calinski-Harabasz Index: 281.4956646348544
Davies-Bouldin Score: 1.1967862406183258

BIRCH:
Silhouette Score: 0.3434275673932862
Calinski-Harabasz Index: 276.65136223283355
Davies-Bouldin Score: 1.1944878069785223

DBSCAN:
Silhouette Score: 0.11259270668373579
Calinski-Harabasz Index: 0.009183203186465221
Davies-Bouldin Score: 220.3932234801588

OPTICS:
Silhouette Score: 0.22407614801798287
Calinski-Harabasz Index: 158.30783381020225
Davies-Bouldin Score: 1.5325864672137106
```

3.Create a synthetic datset with the datasets.make_moons method in sklearn. Apply K-means clustering, BIRCH, DBSCAN and OPTICS to the dataset.

```
print("KMeans:")
print("Silhouette Score:", silhouette_score(X, kmeans_labels))
print("Calinski-Harabasz Index:", calinski_harabasz_score(X, kmeans_labels))
print("Davies-Bouldin Score:", davies_bouldin_score(X, kmeans_labels))

print("\nBIRCH:")
print("Silhouette Score:", silhouette_score(X, birch_labels))
print("Calinski-Harabasz Index:", calinski_harabasz_score(X, birch_labels))
print("Davies-Bouldin Score:", davies_bouldin_score(X, birch_labels))

print("\nDBSCAN:")
dbscan_core_samples = dbscan_labels[dbscan_labels != -1]
dbscan_core_data = X[dbscan_labels != -1]
print("Silhouette Score:", silhouette_score(dbscan_core_data, dbscan_core_samples))
print("Calinski-Harabasz Index:", calinski_harabasz_score(dbscan_core_data, dbscan_core_samples))
print("Davies-Bouldin Score:", davies_bouldin_score(dbscan_core_data, dbscan_core_samples))
```

4. Evaluate the results with the metrics seen the previous day. Which metrics can you use in these datasets? Which algorithm is working better?


```
print("\nOPTICS:")
optics_core_samples = optics_labels[optics_labels != -1]
optics_core_data = X[optics_labels != -1]
print("Silhouette Score:", silhouette_score(optics_core_data, optics_core_samples))
print("Calinski-Harabasz Index:", calinski_harabasz_score(optics_core_data, optics_core_samples))
print("Davies-Bouldin Score:", davies_bouldin_score(optics_core_data, optics_core_samples))
```

```
KMeans:
Silhouette Score: 0.48205688047366435
Calinski-Harabasz Index: 717.0673499842684
Davies-Bouldin Score: 0.7892748095617785

BIRCH:
Silhouette Score: 0.45309474507052294
Calinski-Harabasz Index: 584.7139237411263
Davies-Bouldin Score: 0.7659278313259685

DBSCAN:
Silhouette Score: 0.32744389724050826
Calinski-Harabasz Index: 327.84762425843377
Davies-Bouldin Score: 1.1558314675497603

OPTICS:
Silhouette Score: 0.6201948146154512
Calinski-Harabasz Index: 1433.04805650400075
Davies-Bouldin Score: 0.4881188833752319
```

```
X, y = make_moons(n_samples=500, noise=0.1, random_state=42)

# Plot the dataset
# plt.figure(figsize=(8, 6))
# plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', s=10)
# plt.title("Two Interleaving Moons")
# plt.xlabel("X")
# plt.ylabel("Y")
# plt.show()

kmeans = KMeans(n_clusters=2, random_state=42).fit(X)
kmeans_labels = kmeans.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=kmeans_labels, cmap='viridis', s=10)
plt.title("K-Means Clustering")
plt.show()

birch = Birch(n_clusters=2).fit(X)
birch_labels = birch.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=birch_labels, cmap='viridis', s=10)
plt.title("BIRCH Clustering")
plt.show()

dbscan = DBSCAN(eps=0.2, min_samples=5).fit(X)
dbscan_labels = dbscan.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=dbscan_labels, cmap='viridis', s=10)
plt.title("DBSCAN Clustering")
plt.show()

optics = OPTICS(min_samples=5, xi=0.05, min_cluster_size=0.1).fit(X)
optics_labels = optics.labels_

plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=optics_labels, cmap='viridis', s=10)
plt.title("OPTICS Clustering")
plt.show()
```