Unsupervised Machine Learning with Python

Section 8.1: Metrics for Measuring Quality of Clustering

Quality of Clustering

- "Well Clustered":
  - Points within cluster are close to each other and clusters are well separated

- "Poorly Clustered":
  - Points within cluster may be far apart and clusters close together

Metrics for Measuring Quality of Clustering

- This section covers “Internal” clustering metrics, which are based on clustering results only
- “External” clustering metrics, involve using additional classification information (discussed in Section 10)
- Internal metrics involve computing ratio of distance between points within cluster to distance between clusters
- Davies-Bouldin Index
  - Based on cluster-level quantities
  - Amount of work is $O(M)$ as $M \to \infty$ (M is number of data points)
- Silhouette Index
  - Silhouette score is computed for each data point and then averaged to get index value for dataset
  - Amount of work is $O(M^2)$ as $M \to \infty$
- See also Dunn Index

Davies-Bouldin Index

- Define S_i to denote cluster i and C_i to denote the center of cluster i
- Compactness of cluster i is the average distance between points in cluster i and its centre

$$compact(S_i) = \frac{1}{|S_i|}\sum_{X \in S_i} dist(\,C_i, X)$$

- Distance between clusters is defined as distance between cluster centres: $M_{ij} = dist(C_i, C_j)$
- Define D matrix as

$$D_{ij} = \frac{compact(S_i) + compact(S_j)}{M_{ij}} \quad i \neq j \quad D_{ii} = 0$$

This entry is ratio of compactness for clusters i and j to the distance between them

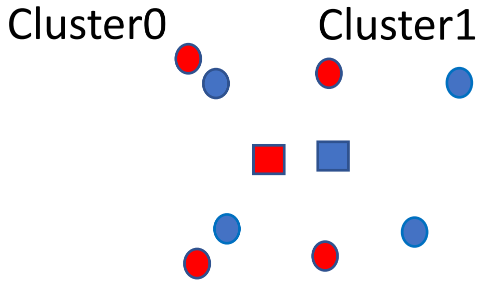
- Davies-Bouldin Index defined (N is number of clusters)

$$DB = \frac{1}{N}\sum_{i} max_j D_{ij}$$

Davies-Bouldin Index Examples

- Davies-Bouldin Index close to 0 indicates well separated, compact clusters
- Davies-Bouldin Index >>1 indicates poorly separated clusters
- Well separated “compact” clusters
  - $compact(Clus_0), compact(Clus_1) < dist(C_0, C_1)$
  - DB Index < 1

- Not well separated clusters not compact clusters
  - $compact(Clus_0), compact(Clus_1) > dist(C_0, C_1)$
  - DB Index >1



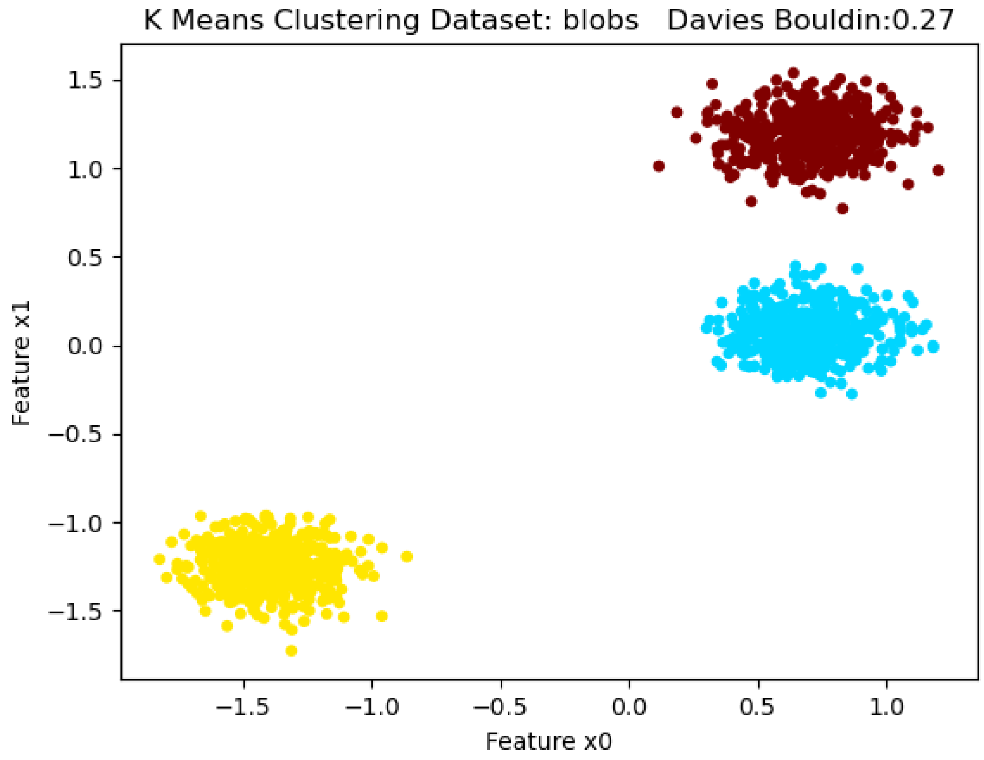
Davies-Bouldin Index Examples

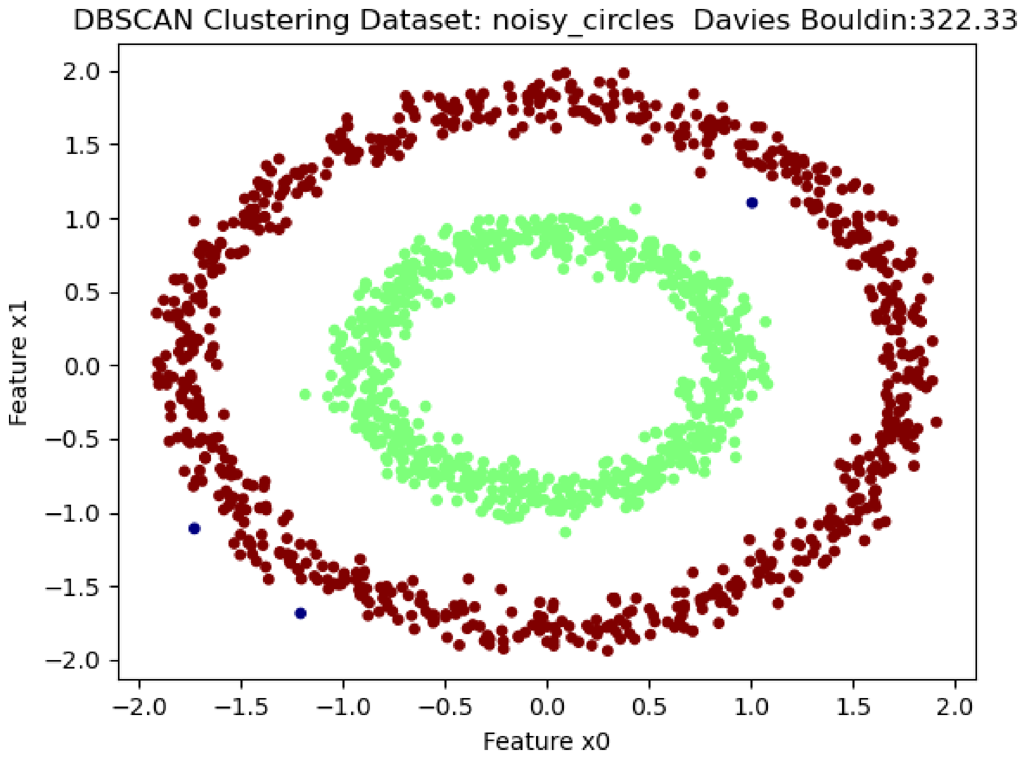
Example 1:

- “blobs” dataset with 1500 points using K Means (3 clusters)

Example 2:

- “noisy_circles” dataset with 1500 points using DBSCAN (minpts = 5, $\varepsilon = 0.18$)





Silhouette Index

- Silhouette index is defined for each point in the dataset and index value for entire dataset is mean of these individual values.
- Silhouette index is between -1 and 1
- Silhouette index is 0 for cluster with 1 point
- For X_i in cluster S_i with more than 1 point, define (avg distance to other points in cluster):

$$a(X_i) = \frac{1}{|S_i| - 1} \sum_{X \in S_i} dist(X_i, X)$$

- Define minimum avg distance to points within other clusters as:

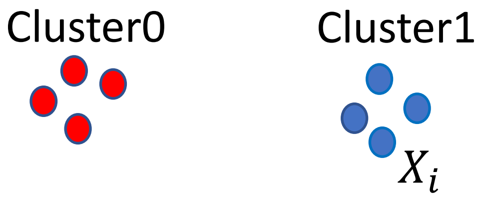
$$b(X_i) = min_{k \neq i} \frac{1}{|S_k|} \sum_{X \in S_k} dist(X_i, X)$$

- Silhouette index for X_i defined as:

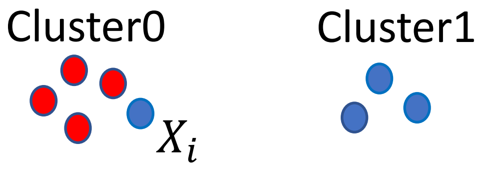
$$Silhouette(X_i) = \frac{b(X_i) - a(X_i)}{\max(a(X_i), b(X_i))}$$

Silhouette Index Examples

- Silhouette index near 1 indicates well separated clusters
- Silhouette index near -1 indicates poorly separated clusters
- Well separated “compact” clusters
  - $a(X_i) \ll b(X_i)$
  - $Silhouette(X_i) = \frac{b(X_i)-a(X_i)}{\max(a(X_i),b(X_i))} \approx 1$



- Not well separated clusters
  - $a(X_i) \gg b(X_i)$
  - $Silhouette(X_i) = \frac{b(X_i)-a(X_i)}{\max(a(X_i),b(X_i))} \approx -1$



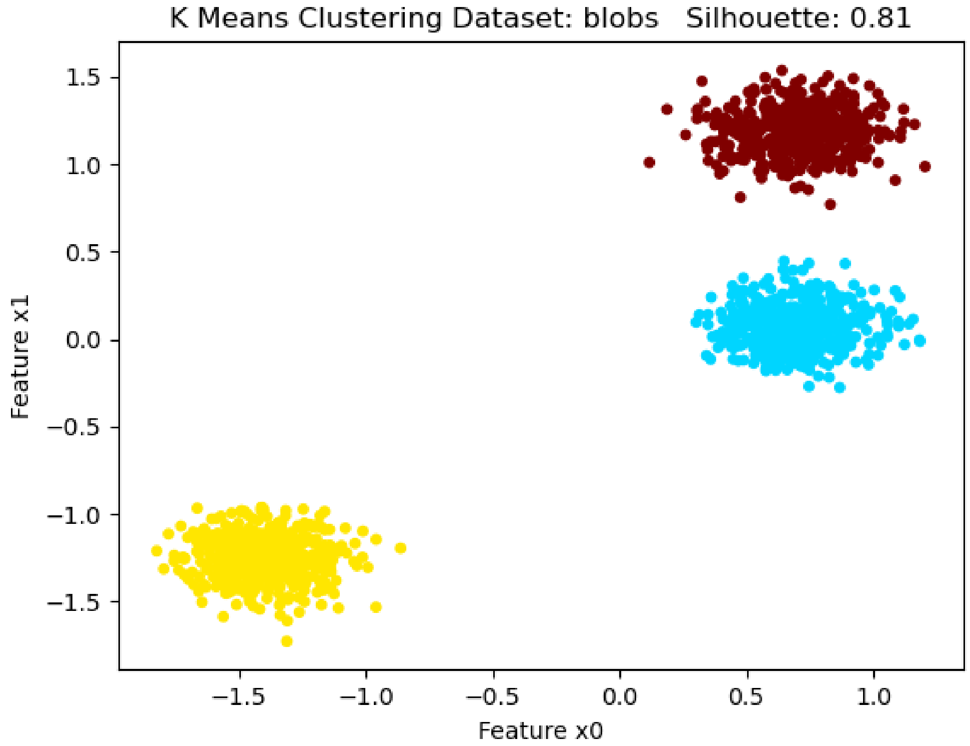
Silhouette Index Examples

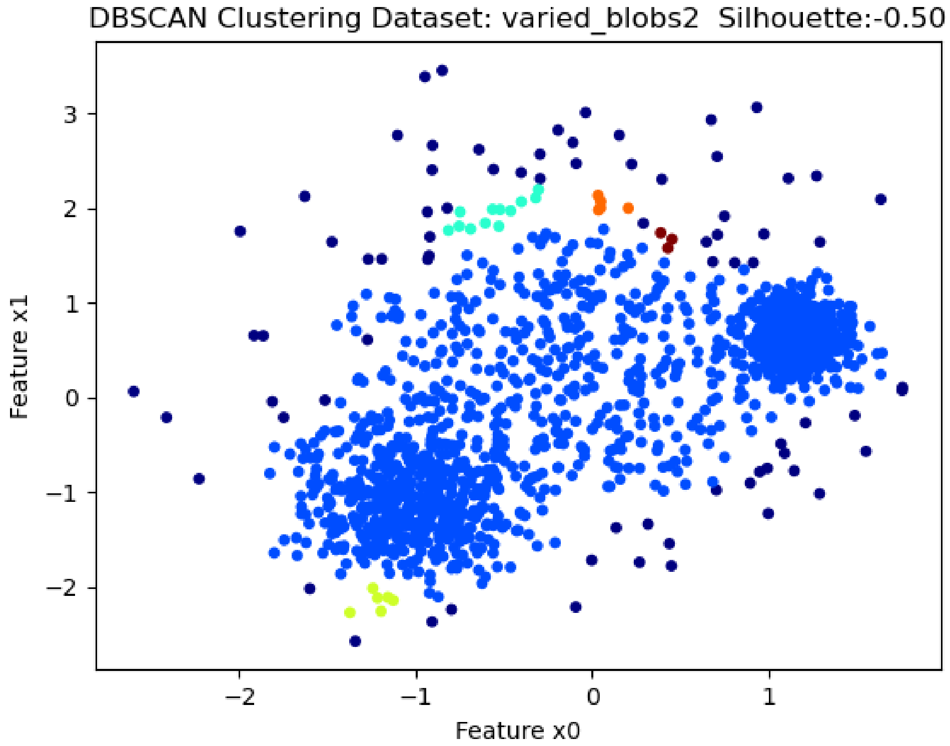
Example 1:

- "blobs" dataset with 1500 points using K Means (3 clusters)

Example 2:

- "varied_blobs2" dataset with 1500 points using DBSCAN (minpts = 5, $\varepsilon = 0.18$)





Implementation Details

- DBSCAN Implementation:
  - Cluster assignment is -1 for all noise points
- Hierarchical Clustering Implementation:
  - Cluster assignment is -1 for all points not yet combined into a cluster
- As each point in these cases is its own cluster need to assign unique label
- In preprocessing step for Davies-Bouldin/Silhouette index calculation, for cluster assignment = -1, re-assign to –(index value +1)
- Example
  - Original assignment: [0,-1,1,0,1,2,2,-1,1,-1]
  - New assignment: [0,-2,1,0,1,2,2,-8,1,-10]

Index=1 Index=7 Index=9

Implementation Details

For Davies Bouldin, need to compute symmetric matrix D:

$$D_{ij} = \frac{compact(S_i) + compact(S_j)}{M_{ij}} \quad i \neq j \quad D_{ii} = 0$$

- Compute upper triangular part of D (example 3x3 case)

$$U = \begin{bmatrix} 0 & D_{01} & D_{02} \\ 0 & 0 & D_{12} \\ 0 & 0 & 0 \end{bmatrix}$$

- $D = U + U^T$

Davies-Bouldin/Silhouette Code Design

Function	Input	Description
davies_bouldin	X (2d numpy array) cluster_assignment (1d numpy array)	Computes Davies-Bouldin index for dataset X given the cluster assignments Return: Davies-Bouldin index
silhouette	X (2d numpy array) cluster_assignment (1d numpy array)	Computes Silhouette index for dataset X given the cluster assignments Return: Silhouette index

Metrics Code Walkthrough

Code located at:

- UnsupervisedML/Code/Programs

Files to Review	Description
metrics.py	Contains functions for computing clustering metrics
driver_kmeans.py	Show example of producing Davies-Bouldin and Silhouette index values

Course Resources at:

- https://github.com/satishchandrareddy/UnsupervisedML/

- Stop video if you would like to implement code yourself first

Unsupervised Machine Learning with Python

Section 8.2: Comparison of Algorithms

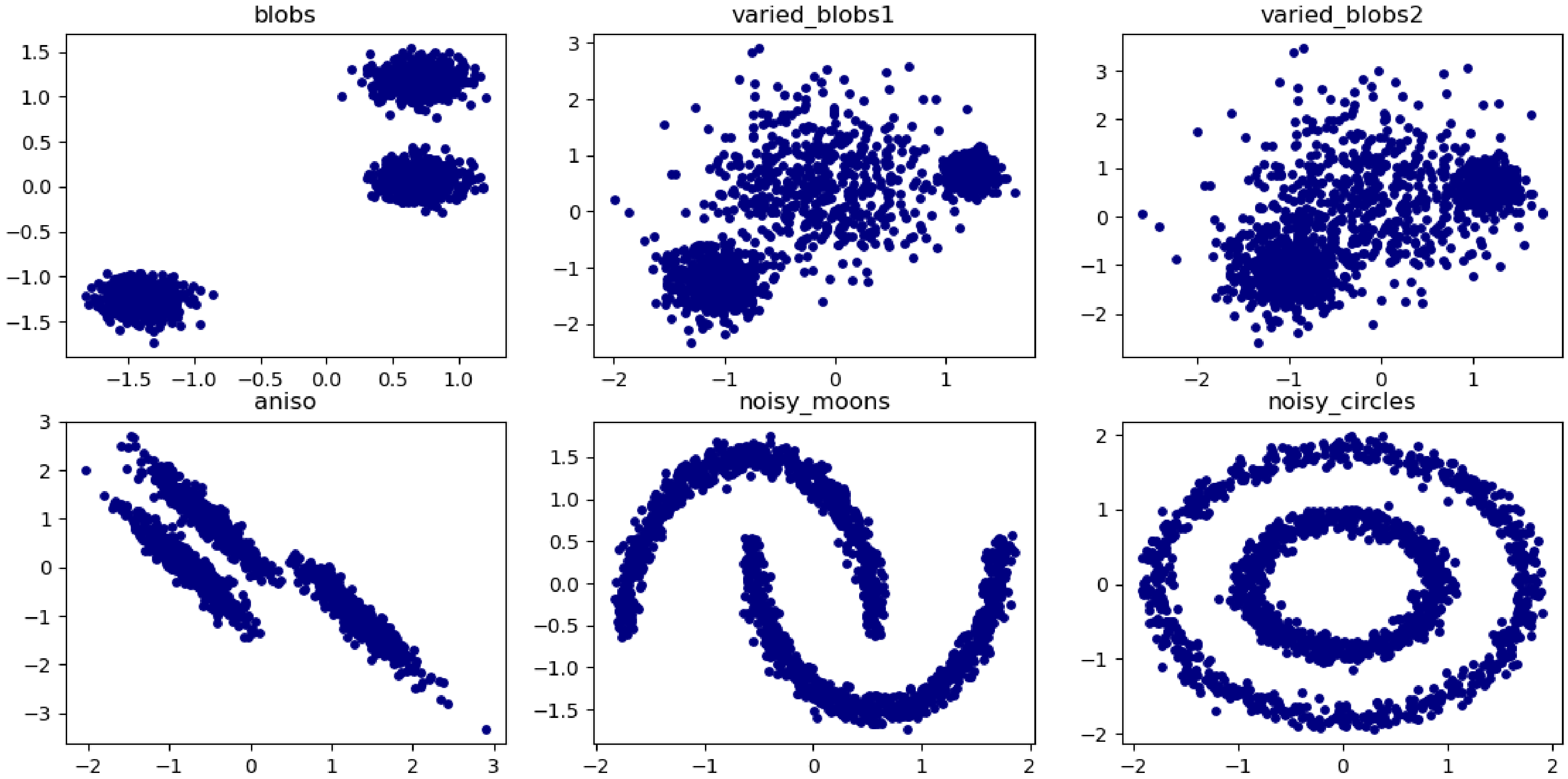
Comparison of Clustering Algorithms

- Compare clustering using K Means, Gaussian Mixture Model and DBSCAN for the 6 sklearn datasets
- Will not use Hierarchical Clustering since it is a impractical choice if there are a large number of data points
  - Amount of work is $O(M^3)$ as $M \to \infty$ (M is number of data points)
  - Amount of work is $O(M)$ for K Means and GMM and $O(M^2)$ for DBSCAN
- Similar to what is done in sklearn

https://scikit-learn.org/stable/modules/clustering.html

Comparison of Algorithms: Datasets

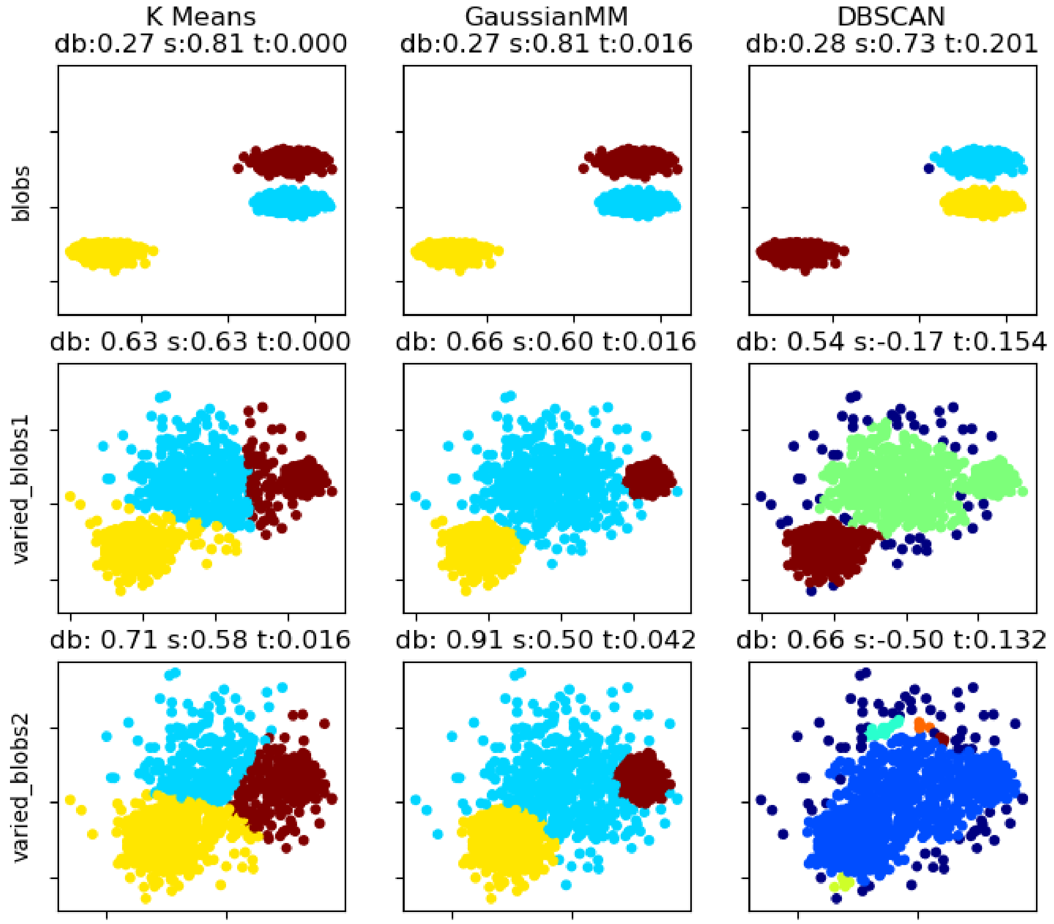
- sklearn datasets using 1500 data points



Comparison of Algorithms: Settings

Dataset/Algorithm	K Means	Gaussian Mixture Model	DBSCAN
blobs	3 clusters, kmeans++	3 clusters, kmeans++	minpts = 5, epsilon = 0.18
varied_blobs1	3 clusters, kmeans++	3 clusters, kmeans++	minpts = 5, epsilon = 0.18
varied_blobs2	3 clusters, kmeans++	3 clusters, kmeans++	minpts = 5, epsilon = 0.18
aniso	3 clusters, kmeans++	3 clusters, kmeans++	minpts = 5, epsilon = 0.18
noisy_moons	2 clusters, kmeans++	2 clusters, kmeans++	minpts = 5, epsilon = 0.18
noisy_circles	2 clusters, kmeans++	2 clusters, kmeans++	minpts = 5, epsilon = 0.18

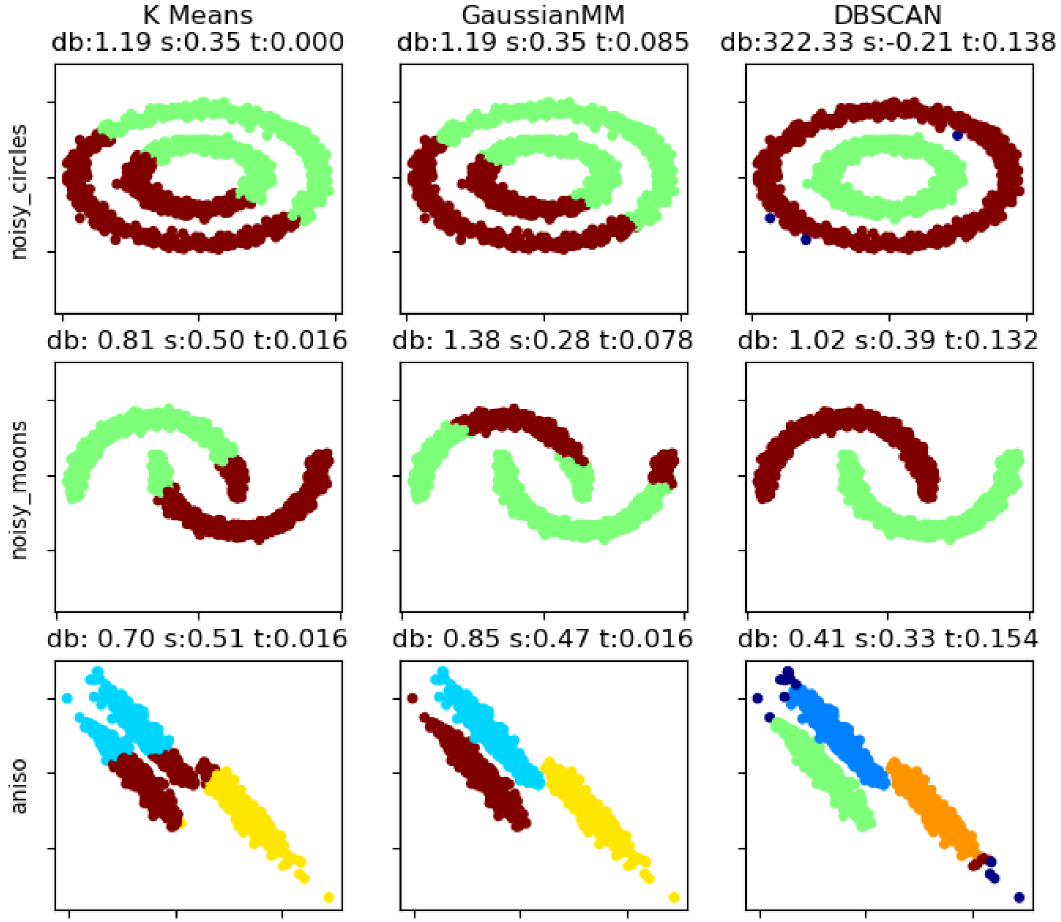
Comparison of Algorithms: Set 1



Notes:

- K Means and GaussianMM:
  - Perform similarly
  - K Means faster than GMM
- DBSCAN: impacted by minpts and epsilon:
  - If density too low: then many points belong to a single cluster
  - If density too high: then lots of noise points
  - Does not do well with clusters of varying density
  - DBSCAN slower than K Means and GMM for these datasets

Comparisons of Algorithms: Set 2



Notes:

- K Means:
  - Does not work well for non-convex regions (circles or moons)
  - Does not work well for elongated regions (aniso)
- GMM:
  - Does not work well for non-convex regions (circles or moons)
  - Can handle elongated regions
- DBSCAN:
  - Can handle non-convex regions

Comparison of Clustering Algorithms

- None of the algorithms (K Means, Gaussian MM, DBSCAN) performs better than the others for all datasets
- Silhouette and Davies-Bouldin Index values give some information, but are not perfect
  - For Silhouette want value to be close to +1
    - For noisy_moons: Silhouette for K Means = 0.50, Silhouette for DBSCAN = 0.39, but DBSCAN has “better” clustering
  - For Davies-Bouldin want value to be close to 0
    - For aniso: Davies-Bouldin for K Means = 0.70, Davies-Bouldin for GMM = 0.85, but GMM has “better”clustering

8.2 Comparison Code Walkthrough

Code located at:

- UnsupervisedML/Code/Programs

Files to Review	Description
driver_comparison.py	Driver for comparing algorithms

Course Resources at:

- https://github.com/satishchandrareddy/UnsupervisedML/

- Stop video if you would like to implement code yourself first