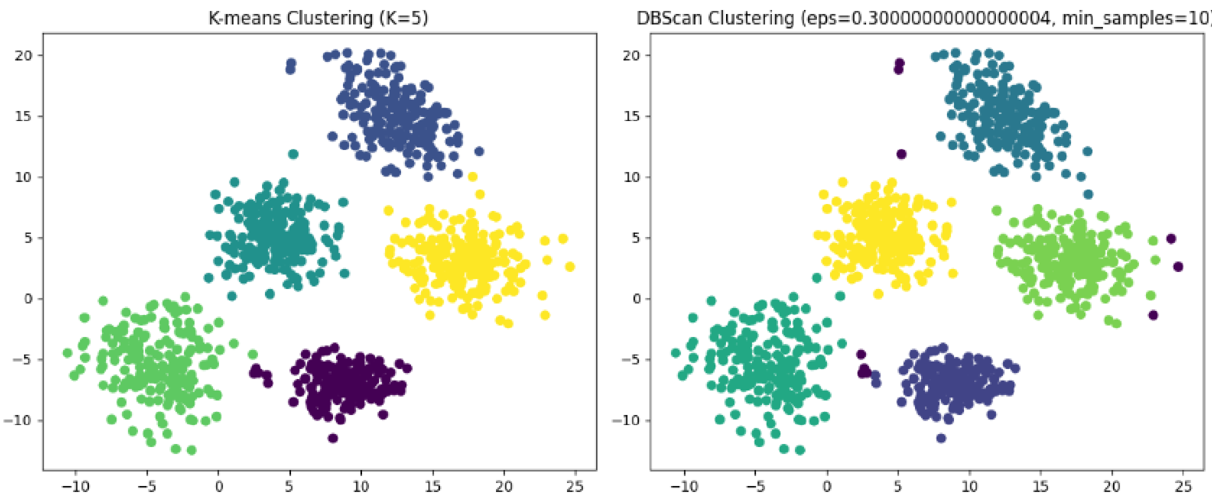
Laurence Kim

COEN 281

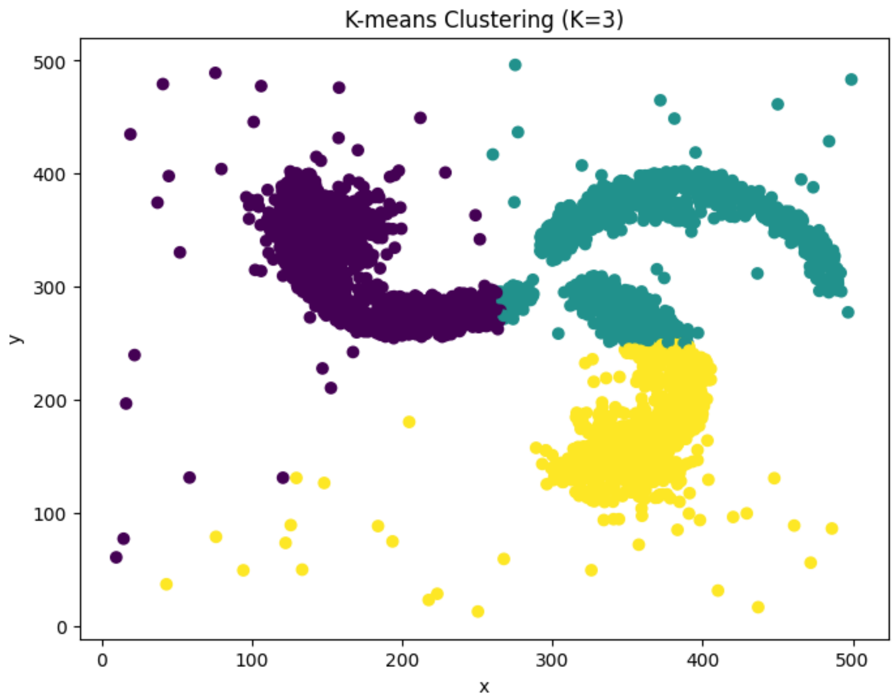
11/14/23

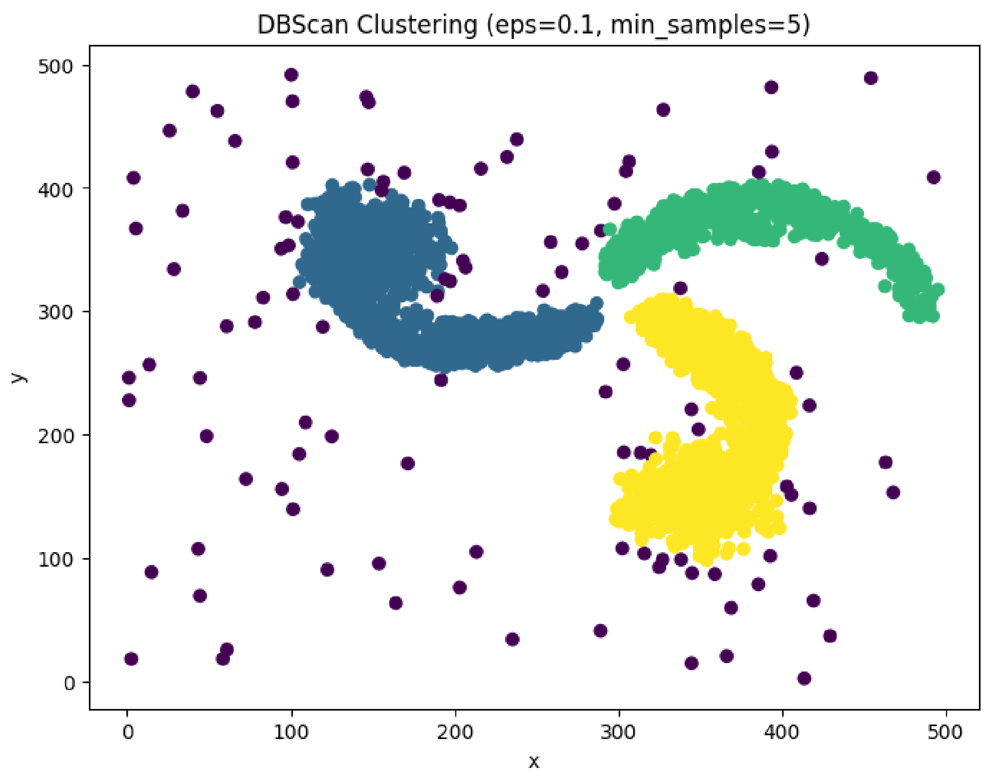
<u>Assignment 3: Cluster Analysis</u>

1. Data set 1
    1.1. K-means and DBScan are different clustering algorithms and their results can vary depending on the dataset and the choice of hyperparameters. K-means typically assumes that clusters are spherical and equally sized. It works well when the clusters are relatively well-separated and have similar shapes and sizes. It assigns every point to a cluster, even if the cluster is not a good fit for the data. DBScan, on the other hand, is density-based and can find clusters of arbitrary shapes. It doesn't require specifying the number of clusters in advance and can identify noise points as well. Understanding the characteristics of these algorithms, we see that points that seem like noise are still attributed to its nearest cluster, while the DBScan algorithm groups all the noise points into one.



2. Data set 2
    2.1. Understanding the difference between k means and dbscan above…I first inferred that using k-means on a spiral dataset with noise would inaccurately capture the underlying spiral structure as it would struggle to handle noise and outliers effectively. This turned out to be true. Furthermore, I assumed that DBSCAN on a spiral dataset with noise, would be very effective in obtaining clusters that accurately capture the spiral pattern and handle noise effectively compared to k-means. This happened to be the case.


K-means Clustering (K=3)
500
400
300
200
100
0
0
100
200
300
400
500
x
y


DBScan Clustering (eps=0.1, min_samples=5)
500
400
300
200
100
0
0
100
200
300
400
500
x
y

Code:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import numpy as np

data = pd.read_csv("size_1000_n_5_sepval_0.2.csv")



import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.model_selection import GridSearchCV

# Load the dataset
data = pd.read_csv("size_1000_n_5_sepval_0.2.csv")

# Select only the 'x' and 'y' columns
X = data[['x', 'y']]

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Define a range of K values to search
k_values = range(2, 11)

# Perform grid search to find the best K for K-means
best_score = -1
best_k = 0

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    labels = kmeans.labels_
    silhouette_avg = silhouette_score(X_scaled, labels)

    if silhouette_avg > best_score:
        best_score = silhouette_avg
        best_k = k

# Fit K-means with the best K value
best_kmeans = KMeans(n_clusters=best_k, n_init=10, random_state=0)
best_kmeans.fit(X_scaled)
best_k_labels = best_kmeans.labels_

# Perform DBScan with grid search for hyperparameters
eps_values = np.arange(0.1, 1.0, 0.1)
min_samples_values = range(2, 11)

best_dbscan = None
best_dbscan_score = -1

for eps in eps_values:
    for min_samples in min_samples_values:
```

```
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        dbscan_labels = dbscan.fit_predict(X_scaled)

        # Calculate the silhouette score (ignoring noise points)
        num_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
        if num_clusters > 1:
            silhouette_avg = silhouette_score(X_scaled, dbscan_labels)
            if silhouette_avg > best_dbscan_score:
                best_dbscan_score = silhouette_avg
                best_dbscan = dbscan

best_dbscan_labels = best_dbscan.labels_

# Plot the results
plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.scatter(X['x'], X['y'], c=best_k_labels, cmap='viridis')
plt.title(f'K-means Clustering (K={best_k})')

plt.subplot(122)
plt.scatter(X['x'], X['y'], c=best_dbscan_labels, cmap='viridis')
plt.title(f'DBScan Clustering (eps={best_dbscan.eps},
min_samples={best_dbscan.min_samples})')

plt.tight_layout()
plt.show()

data2 = pd.read_csv("spirals.csv")

num_noise_points = 100
noise_data = pd.DataFrame({
```

```
    'x': np.random.uniform(low=0, high=500, size=num_noise_points),
    'y': np.random.uniform(low=0, high=500, size=num_noise_points),
    'color': -1
})

data2_with_noise = pd.concat([data2, noise_data], ignore_index=True)
print(data2_with_noise)

# Create two round clusters
center1 = (150, 350)
center2 = (350, 150)
cluster1 = pd.DataFrame({
    'x': np.random.normal(center1[0], 20, size=500),
    'y': np.random.normal(center1[1], 20, size=500),
    'color': 3
})
cluster2 = pd.DataFrame({
    'x': np.random.normal(center2[0], 20, size=500),
    'y': np.random.normal(center2[1], 20, size=500),
    'color': 4
})

data2_with_clusters = pd.concat([data2_with_noise, cluster1, cluster2], ignore_index=True)


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

```python
from sklearn.model_selection import GridSearchCV

X = data2_with_clusters[['x', 'y']]

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

k_values = range(2, 11)
best_score = -1
best_k = 0

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(X_scaled)
    labels = kmeans.labels_
    silhouette_avg = silhouette_score(X_scaled, labels)

    if silhouette_avg > best_score:
        best_score = silhouette_avg
        best_k = k

# Fit K-means with the best K value
best_kmeans = KMeans(n_clusters=best_k, random_state=0)
best_kmeans.fit(X_scaled)
best_k_labels = best_kmeans.labels_

# Plot K-means results
plt.figure(figsize=(8, 6))
plt.scatter(X['x'], X['y'], c=best_k_labels, cmap='viridis')
plt.title(f'K-means Clustering (K={best_k})')
plt.xlabel('x')
```

```python
plt.ylabel('y')
plt.show()

from sklearn.cluster import DBSCAN

# Perform DBScan clustering with hyperparameter settings
eps_values = [0.1, 0.5, 1.0, 1.5]
min_samples_values = [5, 10, 20]

best_dbscan = None
best_dbscan_score = -1

for eps in eps_values:
    for min_samples in min_samples_values:
        dbscan = DBSCAN(eps=eps, min_samples=min_samples)
        dbscan_labels = dbscan.fit_predict(X_scaled)

        # Calculate the silhouette score (ignoring noise points)
        num_clusters = len(set(dbscan_labels)) - (1 if -1 in dbscan_labels else 0)
        if num_clusters > 1:
            silhouette_avg = silhouette_score(X_scaled, dbscan_labels)
            if silhouette_avg > best_dbscan_score:
                best_dbscan_score = silhouette_avg
                best_dbscan = dbscan

best_dbscan_labels = best_dbscan.labels_

# Plot DBScan results
plt.figure(figsize=(8, 6))
plt.scatter(X['x'], X['y'], c=best_dbscan_labels, cmap='viridis')
```

```
plt.title(f'DBScan Clustering (eps={best_dbscan.eps}, min_samples={best_dbscan.min_samples})')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```