Report Analysis HOMEWORK ASSIGNMENT 6

Requirement for the report: - results for each step, including a comparison of the several models - description of the clustering method - description of the quality metrics for the models.

1. We generate synthetic data for analysis and making a model.
   Input data

```
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
```

```
n_samples_1 = 1000
n_samples_2 = 100
centers = [[0.0, 0.0], [-2.0, -2.0]]
cluster_std = [2.0, 1.0]

X1, y1 = make_blobs(n_samples=[n_samples_1, n_samples_2], centers=centers, cluster_std=cluster_std, random_state=0, shuffle=False)
```

```
print(X1, y1)
```

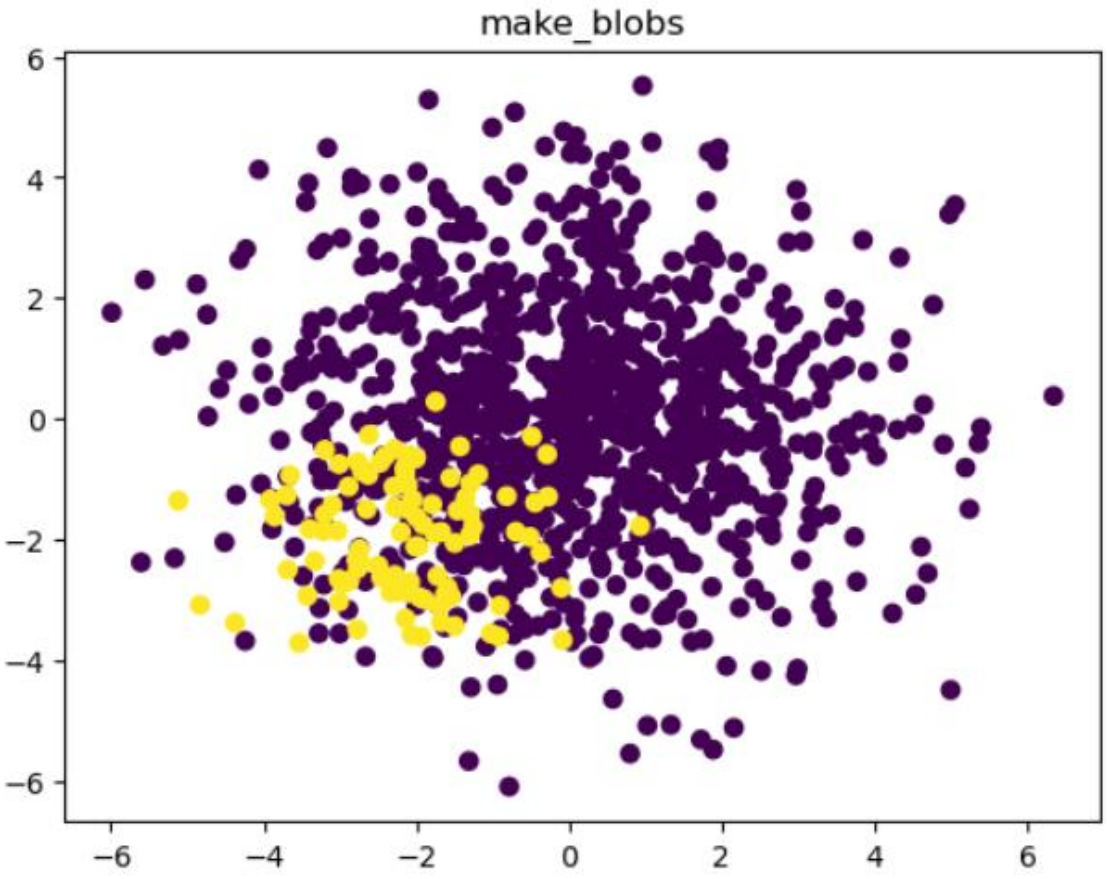
   Output

```
[[ 3.52810469  0.80031442]
 [ 1.95747597  4.4817864 ]
 [ 3.73511598 -1.95455576]
 ...
 [-2.62898071 -0.92992749]
 [-2.6210827  -0.26542783]
 [-3.09828943 -1.42738665]] [0 0 0 ... 1 1 1]
```

2. visualize the input data on the Graph.
   Input data.

```
plt.scatter(X1[:, 0], X1[:, 1], c=y1)
plt.title('make_blobs')
plt.show()
```

   Output.



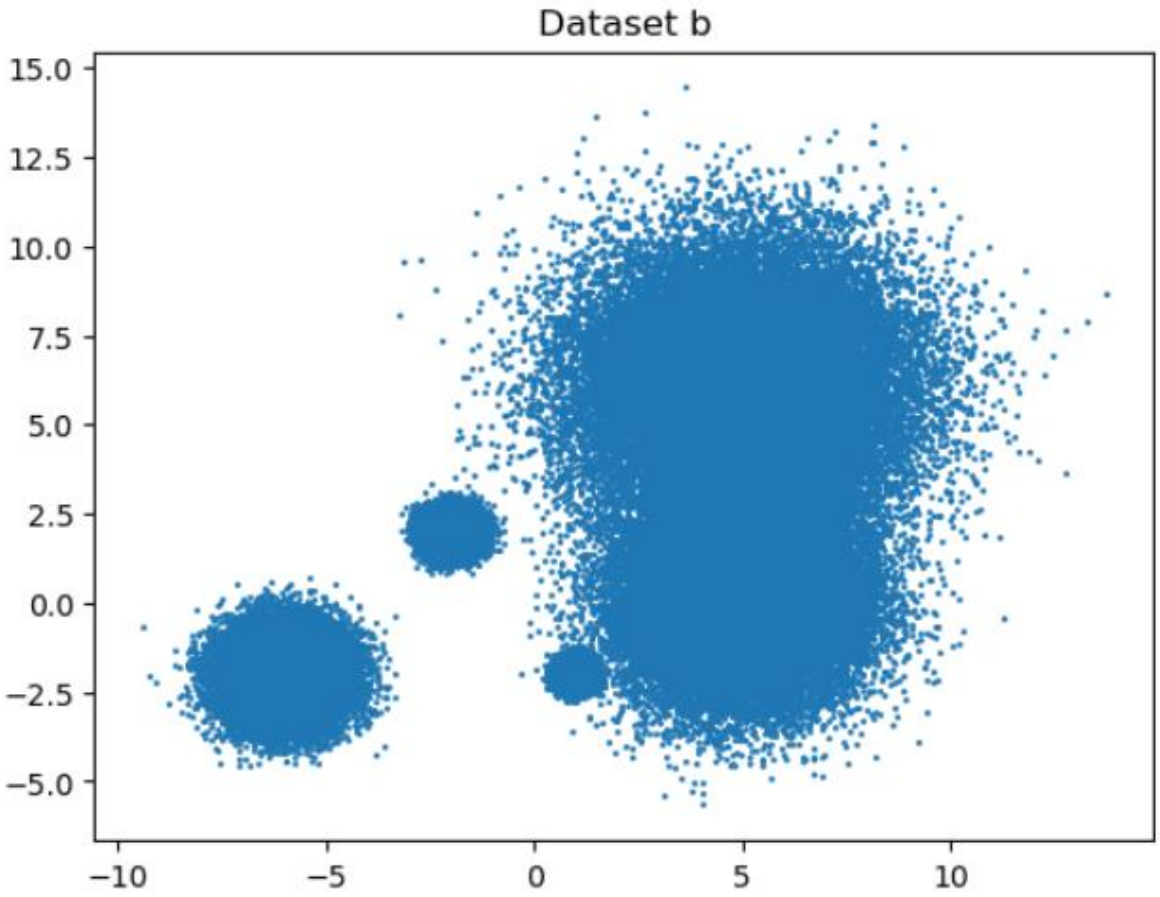
3. For dataset b we will perform as it is step.
   Input data

```
import numpy as np

np.random.seed(0)
n_points_per_cluster = 30000

C1 = [-6, -2] + 0.7 * np.random.randn(n_points_per_cluster, 2)
C2 = [-2, 2] + 0.3 * np.random.randn(n_points_per_cluster, 2)
C3 = [1, -2] + 0.2 * np.random.randn(n_points_per_cluster, 2)
C4 = [4, 4] + 0.1 * np.random.randn(n_points_per_cluster, 2)
C5 = [5, 0] + 1.4 * np.random.randn(n_points_per_cluster, 2)
C6 = [5, 6] + 2.0 * np.random.randn(n_points_per_cluster, 2)

X2 = np.vstack((C1, C2, C3, C4, C5, C6))
```

4. Visualize the dataset b
   Input data.

```
plt.scatter(X2[:, 0], X2[:, 1], s=1)
plt.title('Dataset b')
plt.show()
```

Output



5. Create a clustering model for K-means using: cluster.KMeans and cluster.MiniBatchKMeans

Input data

```
from sklearn.cluster import KMeans, MiniBatchKMeans


kmeans = KMeans(n_clusters=2, random_state=0, n_init=10)
kmeans.fit(X1)

minibatch_kmeans = MiniBatchKMeans(n_clusters=2, random_state=0, n_init=3)
minibatch_kmeans.fit(X1)
```

```
kmeans = KMeans(n_clusters=2, random_state=0, n_init=10)
kmeans.fit(X1)
```

Output

```
MiniBatchKMeans
MiniBatchKMeans(n_clusters=2, n_init=3, random_state=0)
```

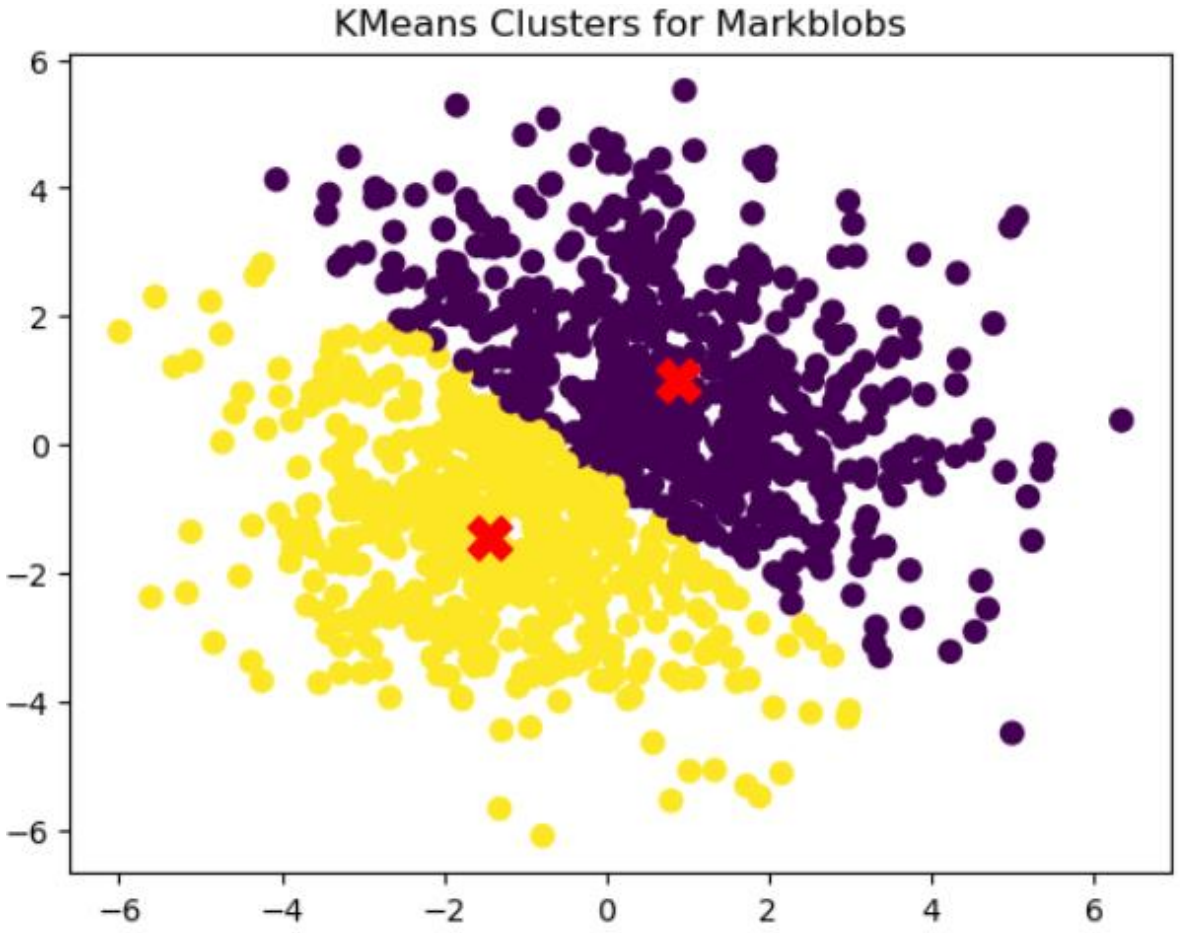
```
KMeans
KMeans(n_clusters=2, n_init=10, random_state=0)
```

6. Visualise through Graph.

   Input data.

```
plt.scatter(X1[:, 0], X1[:, 1], c=y_kmeans_a, s=50, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c='red', marker='X')
plt.title('KMeans Clusters for Markblobs')
plt.show()
```

   Output

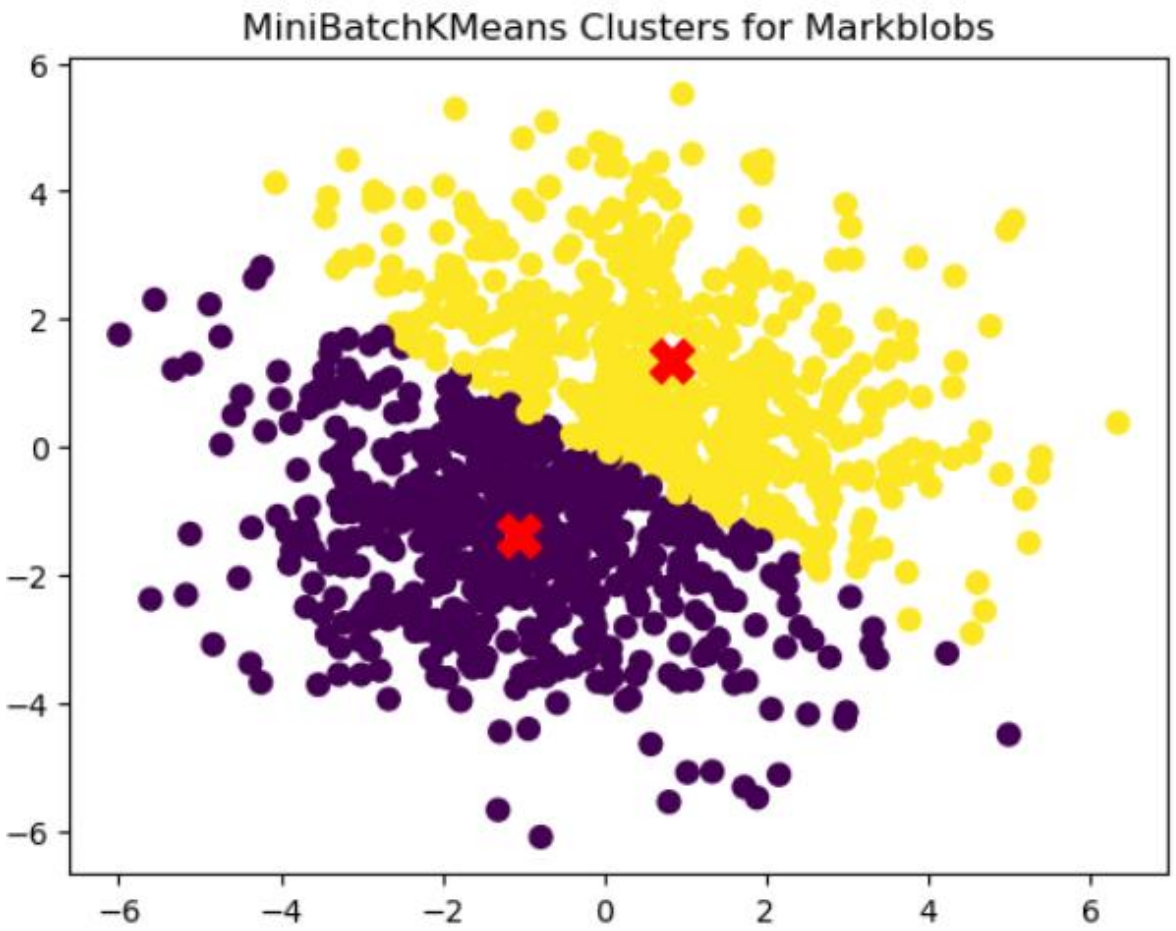


7. Using MiniBatchKMeans

   Input Data

```
plt.scatter(X1[:, 0], X1[:, 1], c=y_minibatch_kmeans_a, s=50, cmap='viridis')
plt.scatter(minibatch_kmeans.cluster_centers_[:, 0], minibatch_kmeans.cluster_centers_[:, 1], s=200, c='red', marker='X')
plt.title('MiniBatchKMeans Clusters for Markblobs')
plt.show()
```

   Output



8. For each alternative model use quality metric from sklearn.metrics: Adjusted Rand Index, Calinski-Harabasz Index, Davies-Bouldin index.

Input data

```
ari_kmeans_a = adjusted_rand_score(y1, y_kmeans_a)
ari_minibatch_kmeans_a = adjusted_rand_score(y1, y_minibatch_kmeans_a)

ch_kmeans_a = calinski_harabasz_score(X1, y_kmeans_a)
ch_minibatch_kmeans_a = calinski_harabasz_score(X1, y_minibatch_kmeans_a)

db_kmeans_a = davies_bouldin_score(X1, y_kmeans_a)
db_minibatch_kmeans_a = davies_bouldin_score(X1, y_minibatch_kmeans_a)
```

```
print(f"KMeans Adjusted Rand Index (Dataset a): {ari_kmeans_a}")
print(f"MiniBatchKMeans Adjusted Rand Index (Dataset a): {ari_minibatch_kmeans_a}")
print(f"KMeans Calinski-Harabasz Index (Dataset a): {ch_kmeans_a}")
print(f"MiniBatchKMeans Calinski-Harabasz Index (Dataset a): {ch_minibatch_kmeans_a}")
print(f"KMeans Davies-Bouldin Index (Dataset a): {db_kmeans_a}")
print(f"MiniBatchKMeans Davies-Bouldin Index (Dataset a): {db_minibatch_kmeans_a}")
```

Output.

```
KMeans Adjusted Rand Index (Dataset a): 0.04505812028448698
MiniBatchKMeans Adjusted Rand Index (Dataset a): 0.012162992826630218
KMeans Calinski-Harabasz Index (Dataset a): 633.0812630057959
MiniBatchKMeans Calinski-Harabasz Index (Dataset a): 621.2394181691559
KMeans Davies-Bouldin Index (Dataset a): 1.1381760122131444
MiniBatchKMeans Davies-Bouldin Index (Dataset a): 1.1551090815484246
```

For dataset b we will perform the same step.

Input Data.

```
kmeans_b = KMeans(n_clusters=6, random_state=0, n_init=10)
kmeans_b.fit(X2)
y_kmeans_b = kmeans_b.predict(X2)

minibatch_kmeans_b = MiniBatchKMeans(n_clusters=6, random_state=0, n_init=3)
minibatch_kmeans_b.fit(X2)
y_minibatch_kmeans_b = minibatch_kmeans_b.predict(X2)
```

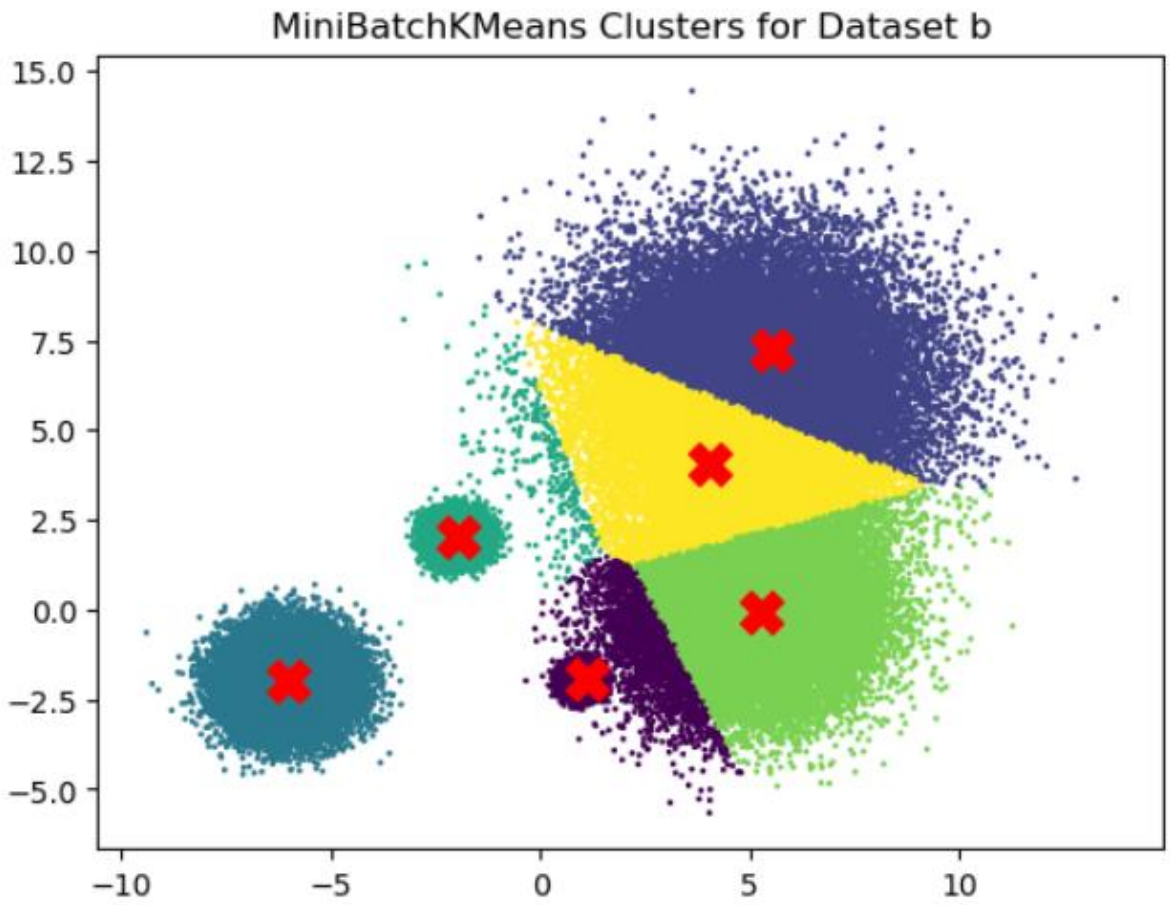
```
plt.scatter(X2[:, 0], X2[:, 1], c=y_kmeans_b, s=1, cmap='viridis')
plt.scatter(kmeans_b.cluster_centers_[:, 0], kmeans_b.cluster_centers_[:, 1], s=200, c='red', marker='X')
plt.title('KMeans Clusters for Dataset b')
plt.show()
```

Output.

Input data.

```
plt.scatter(X2[:, 0], X2[:, 1], c=y_minibatch_kmeans_b, s=1, cmap='viridis')
plt.scatter(minibatch_kmeans_b.cluster_centers_[:, 0], minibatch_kmeans_b.cluster_centers_[:, 1], s=200, c='red', marker='X')
plt.title('MiniBatchKMeans Clusters for Dataset b')
plt.show()
```

Output



Metrics Analysis.

Input Data.

```
ari_kmeans_b = adjusted_rand_score(y_kmeans_b, y_kmeans_b)
ari_minibatch_kmeans_b = adjusted_rand_score(y_minibatch_kmeans_b, y_minibatch_kmeans_b)

ch_kmeans_b = calinski_harabasz_score(X2, y_kmeans_b)
ch_minibatch_kmeans_b = calinski_harabasz_score(X2, y_minibatch_kmeans_b)

db_kmeans_b = davies_bouldin_score(X2, y_kmeans_b)
db_minibatch_kmeans_b = davies_bouldin_score(X2, y_minibatch_kmeans_b)

print(f"KMeans Adjusted Rand Index (Dataset b): {ari_kmeans_b}")
print(f"MiniBatchKMeans Adjusted Rand Index (Dataset b): {ari_minibatch_kmeans_b}")
print(f"KMeans Calinski-Harabasz Index (Dataset b): {ch_kmeans_b}")
print(f"MiniBatchKMeans Calinski-Harabasz Index (Dataset b): {ch_minibatch_kmeans_b}")
print(f"KMeans Davies-Bouldin Index (Dataset b): {db_kmeans_b}")
print(f"MiniBatchKMeans Davies-Bouldin Index (Dataset b): {db_minibatch_kmeans_b}")
```

Output

```
KMeans Adjusted Rand Index (Dataset b): 1.0
MiniBatchKMeans Adjusted Rand Index (Dataset b): 1.0
KMeans Calinski-Harabasz Index (Dataset b): 575061.5806580335
MiniBatchKMeans Calinski-Harabasz Index (Dataset b): 575048.5381234039
KMeans Davies-Bouldin Index (Dataset b): 0.5056380725254451
MiniBatchKMeans Davies-Bouldin Index (Dataset b): 0.5056516255788712
```

Description of quality metrics and comparision of the model and analysis.

Dataset a

Model	ARI	CH	DB
K Means	0.04	633.0	1.13
MiniBatch KMeans	0.012	621.0	1.155

KMeans ARI = 0.04: Indicates a low level of similarity between the true labels and the predicted labels.

MiniBatchKMeans ARI = 0.012: Even lower similarity between true and predicted labels compared to KMeans.

Calinski-Harabasz Index (CH)

KMeans CH = 633.0: Higher value indicates well-defined clusters.

MiniBatchKMeans CH = 621.0: Slightly lower than KMeans, indicating slightly less well-defined clusters.

Davies-Bouldin Index (DB)

KMeans has a slightly lower DB score, suggesting better separation between clusters compared to MiniBatchKMeans.

KMeans slightly outperforms MiniBatchKMeans in terms of ARI, CH, and DB scores, indicating better clustering quality for Dataset a.

Dataset b

Model	ARI	CH	DB
K Means	1.0	575061.58065	0.50563
MiniBatchKMeans	1.0	575048.5381	0.5065

KMeans vs. MiniBatchKMeans: Both models perform almost identically, with KMeans slightly outperforming MiniBatchKMeans in terms of CH

Calinski-Harabasz Index (CH)and DB scores.KMeans slightly outperform than MiniBatchKMeans.

Adjusted Rand Index (ARI) perfact in both the cases.

MiniBatchKMeans DB = 0.5065: Slightly higher than KMeans, but still low, indicating good separation between clusters.

Overall for both the dataset KMeans outperform than the MiniBatchKMeans also this indicates for smaller dataset KMeans is better.