Clustering Algorithm Performance Evaluation

In [257]:
```
#importing libraries
import numpy as np
import matplotlib
from matplotlib import pyplot
```

(a) *Import and plot the data and if necessary preprocess it using sklearn.preprocessing*

In [40]:
```
# loading the four datasets using np.loadtxt

ds_1 = np.loadtxt(open("dataset1_c7.csv", "rb"), delimiter = ",", skiprows
ds_2 = np.loadtxt(open("dataset2_c6.csv", "rb"), delimiter = ",", skiprows
ds_3 = np.loadtxt(open("dataset3_c2.csv", "rb"), delimiter = ",", skiprows
ds_4 = np.loadtxt(open("dataset4_c2.csv", "rb"), delimiter = ",", skiprows
```

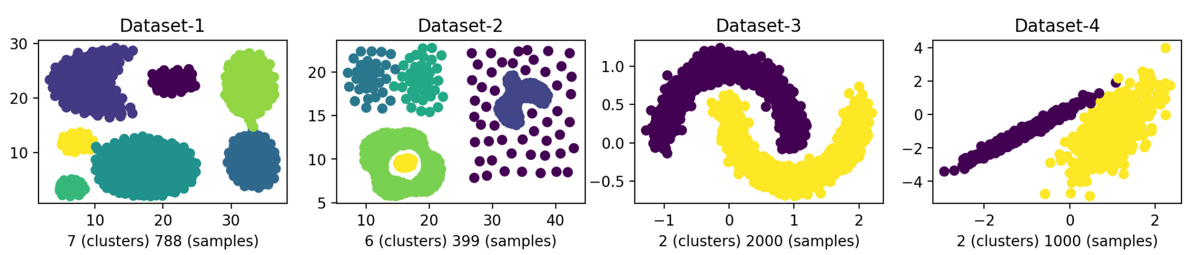
In [258]:
```
# Now, we can visualize these datasets
fig, (plot_1, plot_2, plot_3, plot_4) = pyplot.subplots(1, 4, figsize = (1
plot_1.scatter(ds_1[:,0:1], ds_1[:,1:2], c = ds_1[:,-1])
plot_1.set_title('Dataset-1')
plot_1.set_xlabel("7 (clusters) " + str(len(ds_1)) + " (samples)")

plot_2.scatter(ds_2[:,0:1], ds_2[:,1:2], c = ds_2[:,-1])
plot_2.set_title('Dataset-2')
plot_2.set_xlabel("6 (clusters) " + str(len(ds_2)) + " (samples)")

plot_3.scatter(ds_3[:,0:1], ds_3[:,1:2], c = ds_3[:,-1])
plot_3.set_title('Dataset-3')
plot_3.set_xlabel("2 (clusters) " + str(len(ds_3)) + " (samples)")

plot_4.scatter(ds_4[:,0:1], ds_4[:,1:2], c = ds_4[:,-1])
plot_4.set_title('Dataset-4')
plot_4.set_xlabel("2 (clusters) " + str(len(ds_4)) + " (samples)")
```

Out[258]:
```
Text(0.5, 0, '2 (clusters) 1000 (samples)')
```



(b) *Try four different clustering techniques such as DBSCAN, KMeans, Expectation Maximization (EM), and Average Link, which are already implemented in scikit-learn.*

In [115]:
```
# Importing libraries with different clustering techniques and metrices
from sklearn.cluster import KMeans
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from sklearn import metrics
```

1- DBSCAN

In DBSCAN two parameters are used:

1. eps (epsilon) = max distance b/w two samples or radius
2. min_samples = minimum no. of neighboring samples

By using these two parameters we can visualize the clusters. I followed the documentation for better understanding and implementing DBSCAN on our datasets

[https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html)](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html)

In [125]:

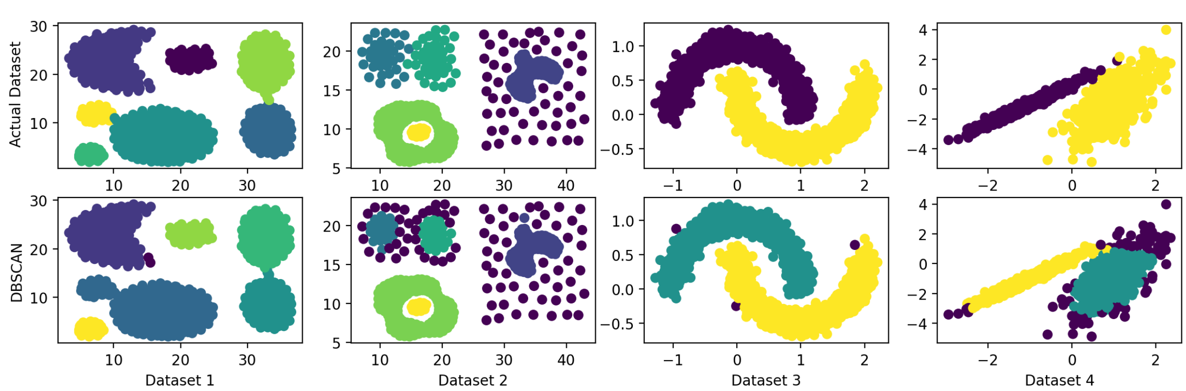
```
# Implementing DBSCAN on datasets
db_1 = DBSCAN(eps = 2, min_samples = 13).fit(ds_1[:,0:2])
db_2 = DBSCAN(eps = 1.5, min_samples = 10).fit(ds_2[:,0:2])
db_3 = DBSCAN(eps = 0.12, min_samples = 3).fit(ds_3[:,0:2])
db_4 = DBSCAN(eps = 0.30, min_samples = 13).fit(ds_4[:,0:2])
```

In [126]:

```
# Visualize the original dataset and with DBSCAN algorithm
fig, (row_1, row_2) = pyplot.subplots(2, 4, figsize = (14,4), dpi = 200)
#comparison of orginal four datset and with DBSCAN
# row_1 is for original data
row_1[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = ds_1[:,-1])
row_1[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = ds_2[:,-1])
row_1[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = ds_3[:,-1])
row_1[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = ds_4[:,-1])
#row_2 is for DBSCAN
row_2[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = db_1.labels_)
row_2[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = db_2.labels_)
row_2[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = db_3.labels_)
row_2[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = db_4.labels_)
#setting y labels
row_1[0].set_ylabel(" Actual Dataset")
row_2[0].set_ylabel("DBSCAN")
#setting x label for each dataset
row_2[0].set_xlabel("Dataset 1")
row_2[1].set_xlabel("Dataset 2")
row_2[2].set_xlabel("Dataset 3")
row_2[3].set_xlabel("Dataset 4")
```

Out[126]:

```
Text(0.5, 0, 'Dataset 4')
```



2. K-Means

In K-Means two parameters are used:

1. n_clusters = numbers of clusters to generate
2. random_state= generation of random number for centroids

By using these two parameters we can observe that the clusters of high density are seperated from the clusters with low densities.I followed the documentation for better understanding and implementing K-Means on our datasets

[https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans)](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans)

In [129]:

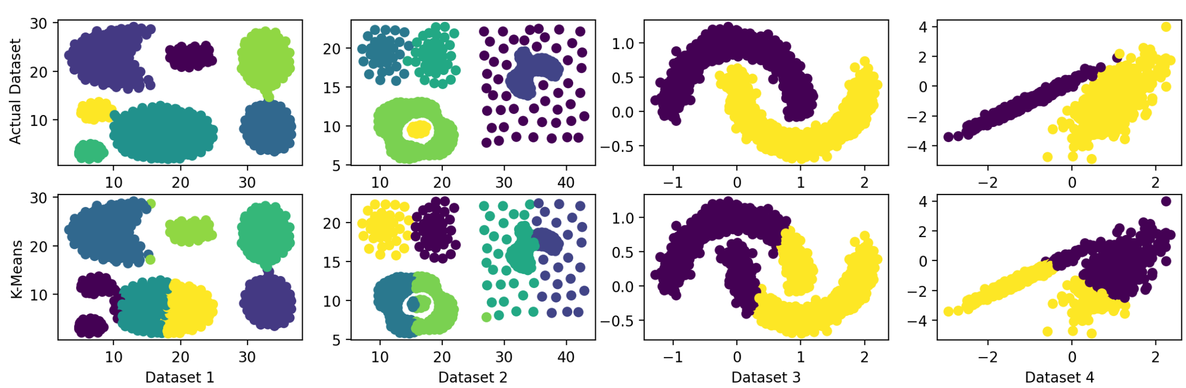
```
# implement K-Means on dataset 1
km_1 = KMeans(n_clusters = 7,random_state = None).fit(ds_1[:,0:2])
km_2 = KMeans(n_clusters = 6,random_state = None).fit(ds_2[:,0:2])
km_3 = KMeans(n_clusters = 2,random_state = None).fit(ds_3[:,0:2])
km_4 = KMeans(n_clusters = 2,random_state = None).fit(ds_4[:,0:2])
```

In [130]:

```
# Visualize the original dataset and withK-Means algorithm
fig, (row_1, row_2) = pyplot.subplots(2, 4, figsize = (14,4), dpi = 200)
#comparison of orginal four datset and with K-means
# row_1 is for original data
row_1[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = ds_1[:,-1])
row_1[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = ds_2[:,-1])
row_1[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = ds_3[:,-1])
row_1[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = ds_4[:,-1])
#row_2 is for K-Means
row_2[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = km_1.labels_)
row_2[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = km_2.labels_)
row_2[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = km_3.labels_)
row_2[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = km_4.labels_)
#setting y labels
row_1[0].set_ylabel(" Actual Dataset")
row_2[0].set_ylabel("K-Means")
#setting x label for each dataset
row_2[0].set_xlabel("Dataset 1")
row_2[1].set_xlabel("Dataset 2")
row_2[2].set_xlabel("Dataset 3")
row_2[3].set_xlabel("Dataset 4")
```

Out[130]:

```
Text(0.5, 0, 'Dataset 4')
```



3. Expectation Minimization (EM)

In Expectation Minimization only one parameters is used:

1. n_components = number of mixture components

By using these two parameters we can observe the probability distribution. I followed the documentation for better understanding and implementing Expectation Minimization on our datasets

[https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html (https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html)](https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html)

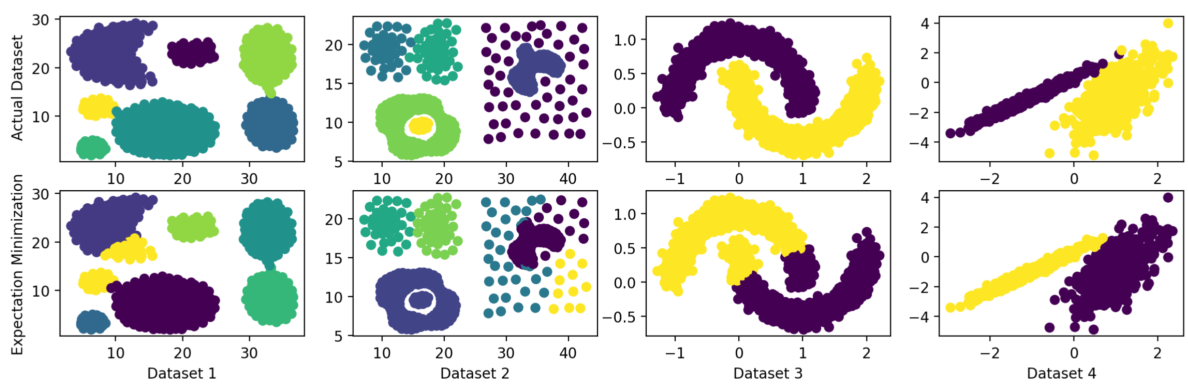
In [109]:

```
# Implementing Expectation Minimization to our datsets
em_1 = GaussianMixture(n_components = 7).fit_predict(ds_1[:,0:2])
em_2 = GaussianMixture(n_components = 6).fit_predict(ds_2[:,0:2])
em_3 = GaussianMixture(n_components = 2).fit_predict(ds_3[:,0:2])
em_4 = GaussianMixture(n_components = 2).fit_predict(ds_4[:,0:2])
```

In [120]:

```
# Visualize the original dataset and with Expectation Minimization algorit
fig, (row_1, row_2) = pyplot.subplots(2, 4, figsize = (14,4), dpi = 200)
#comparison of orginal four datset and with EM
# row_1 is for original data
row_1[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = ds_1[:,-1])
row_1[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = ds_2[:,-1])
row_1[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = ds_3[:,-1])
row_1[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = ds_4[:,-1])
#row_2 is for EM
row_2[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = em_1)
row_2[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = em_2)
row_2[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = em_3)
row_2[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = em_4)
#setting y labels
row_1[0].set_ylabel(" Actual Dataset")
row_2[0].set_ylabel("Expectation Minimization")
#setting x label for each dataset
row_2[0].set_xlabel("Dataset 1")
row_2[1].set_xlabel("Dataset 2")
row_2[2].set_xlabel("Dataset 3")
row_2[3].set_xlabel("Dataset 4")
```

Out[120]: Text(0.5, 0, 'Dataset 4')



4. Average Link

In agglomerative clustering two parameters are used:

1. n_clusters = number of clusters to find
2. linkage = average distange of the observations

By using these two parameters we can observe that clusters are merged for samples. I followed the documentation for better understanding and implementing agglomerative clustering on our datasets

[https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html)](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html)

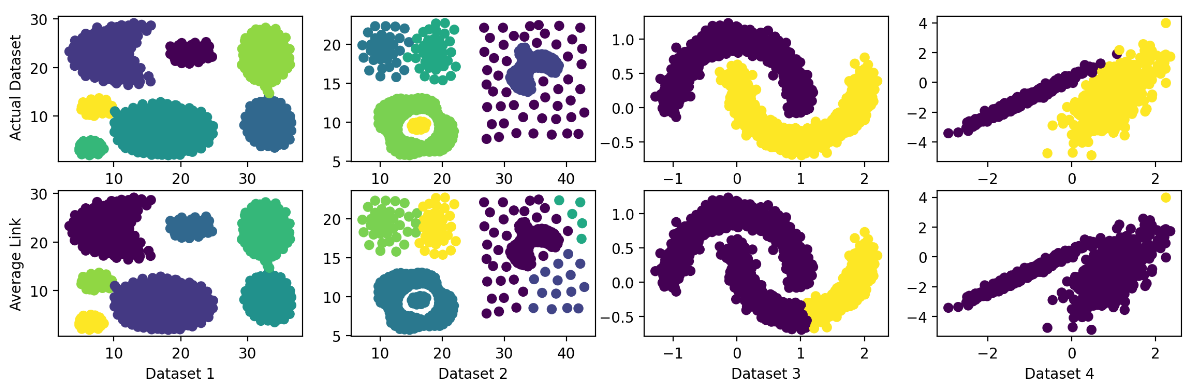
In [121]:

```
# Implementing agglomerative clustering to our datsets
ac_1 = AgglomerativeClustering(n_clusters = 7,linkage = "average").fit(ds_
ac_2 = AgglomerativeClustering(n_clusters = 6,linkage = "average").fit(ds_
ac_3 = AgglomerativeClustering(n_clusters = 2,linkage = "average").fit(ds_
ac_4 = AgglomerativeClustering(n_clusters = 2,linkage = "average").fit(ds_

```

In [122]:
```python
# Visualize the original dataset and with Agglomerative clustering
fig, (row_1, row_2) = pyplot.subplots(2, 4, figsize = (14,4), dpi = 200)
#comparison of orginal four datset and with agglomerative clustering
# row_1 is for original data
row_1[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = ds_1[:,-1])
row_1[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = ds_2[:,-1])
row_1[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = ds_3[:,-1])
row_1[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = ds_4[:,-1])
#row_2 is for EM
row_2[0].scatter(ds_1[:,0:1], ds_1[:,1:2], c = ac_1.labels_)
row_2[1].scatter(ds_2[:,0:1], ds_2[:,1:2], c = ac_2.labels_)
row_2[2].scatter(ds_3[:,0:1], ds_3[:,1:2], c = ac_3.labels_)
row_2[3].scatter(ds_4[:,0:1], ds_4[:,1:2], c = ac_4.labels_)
#setting y labels
row_1[0].set_ylabel(" Actual Dataset")
row_2[0].set_ylabel("Average Link")
#setting x label for each dataset
row_2[0].set_xlabel("Dataset 1")
row_2[1].set_xlabel("Dataset 2")
row_2[2].set_xlabel("Dataset 3")
row_2[3].set_xlabel("Dataset 4")
```

Out[122]: Text(0.5, 0, 'Dataset 4')



Findings: According to the observation made by these visualizations the DBSCAN perform better than other clustering techniques

(b) *Evaluate each clustering technique using Normalized Mutual Information as well as the (Adjusted) Rand Score. For the evaluation, use the sklearn.metrics package*

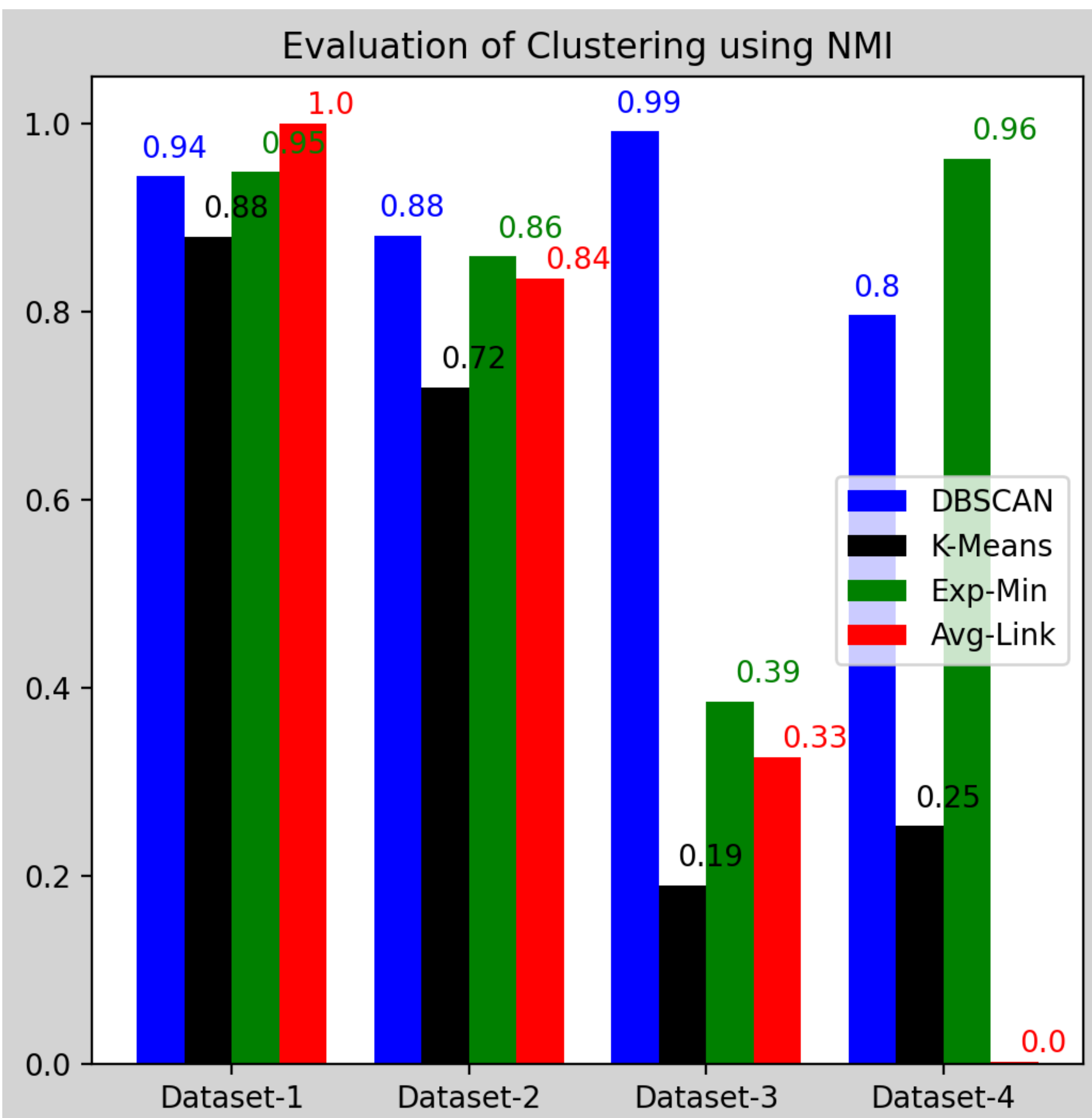
In [143]:
```python
#evaluating our clustering techniques by using normalized mutual informati
#we have used sklearn.metrices .cluster pacakge for the evaluation
from sklearn.metrics.cluster import normalized_mutual_info_score
norm = {"DBSCAN":[], "K-Means":[], "Expected_Min":[], "Avg_Link":[]}
#NMI for DB-Scan on all datasets
norm["DBSCAN"].append(normalized_mutual_info_score(ds_1[:,-1], db_1.labels
norm["DBSCAN"].append(normalized_mutual_info_score(ds_2[:,-1], db_2.labels
norm["DBSCAN"].append(normalized_mutual_info_score(ds_3[:,-1], db_3.labels
norm["DBSCAN"].append(normalized_mutual_info_score(ds_4[:,-1], db_4.labels
# NMI for K-Means on all datasets
norm["K-Means"].append(normalized_mutual_info_score(ds_1[:,-1], km_1.label
norm["K-Means"].append(normalized_mutual_info_score(ds_2[:,-1], km_2.label
norm["K-Means"].append(normalized_mutual_info_score(ds_3[:,-1], km_3.label
norm["K-Means"].append(normalized_mutual_info_score(ds_4[:,-1], km_4.label
#NMI for Expected Minimization on all datasets
norm["Expected_Min"].append(normalized_mutual_info_score(ds_1[:,-1], em_1)
norm["Expected_Min"].append(normalized_mutual_info_score(ds_2[:,-1], em_2)
norm["Expected_Min"].append(normalized_mutual_info_score(ds_3[:,-1], em_3)
norm["Expected_Min"].append(normalized_mutual_info_score(ds_4[:,-1], em_4)
#NMI for Average Link on all datasets
norm["Avg_Link"].append(normalized_mutual_info_score(ds_1[:,-1], ac_1.labe
norm["Avg_Link"].append(normalized_mutual_info_score(ds_2[:,-1], ac_2.labe
norm["Avg_Link"].append(normalized_mutual_info_score(ds_3[:,-1], ac_3.labe
norm["Avg_Link"].append(normalized_mutual_info_score(ds_4[:,-1], ac_4.labe
```

In [255]:

```
# defining function to plot data for finding NMI Score
def nmi(visualization, heading):
    # Plotting the evaluation data
    fig, plt = pyplot.subplots(1, figsize = (6,6), dpi = 200)
    width = 0.2 #plotting bar graph
    plt.bar(np.arange(4) - 1.5*width, visualization["DBSCAN"], width, labe
    plt.bar(np.arange(4) - 0.5*width, visualization["K-Means"], width, lab
    plt.bar(np.arange(4) + 0.5*width, visualization["Expected_Min"], width
    plt.bar(np.arange(4) + 1.5*width, visualization["Avg_Link"], width, la
    #enumerate the NMI score and display the score on plot
    for p, val in enumerate(visualization["DBSCAN"]):
        plt.text(p - 1.9*width, val + 0.02, str(round(val,2)), rotation =
    for p, val in enumerate(visualization["K-Means"]):
        plt.text(p - 0.6*width, val + 0.02, str(round(val,2)), rotation =
    for p, val in enumerate(visualization["Expected_Min"]):
        plt.text(p + 0.6*width, val + 0.02, str(round(val,2)), rotation =
    for p, val in enumerate(visualization["Avg_Link"]):
        plt.text(p + 1.6*width, val + 0.01, str(round(val,2)), rotation =
    # styling the plots
    fig.set_facecolor('lightgray')
    #adding legends to check the categories of clustering techniques
    plt.legend(loc = "center right")
    plt.set_title(heading)
    plt.set_xticks(np.arange(4), ["Dataset-1", "Dataset-2", "Dataset-3", "
```

In [256]:

```
#display the NMI score for each clustering technique on the plot
nmi(norm, "Evaluation of Clustering using NMI")
```



Findings:

1. On Dataset-1 the "Average Link" performed better than "DBSCAN" and "K-Mean" performance is not good
2. On Dataset-2, On Dataset-3, On Dataset-4 "DBSCAN" performed better but "K-Mean" remained worst for dataset 2 and 3 and "Average Link" performed worst for Dataset-4

(d) *why the clustering methods succeed or fail ?*

According to my observation through these visualizations:

1. The DBSCAN algorithm labels the datasets in best way. It has some problems with those samples that are on greater distances or sparse clusters
2. K-Means has issue with those clusters that shows non-convex or have short distances.
3. Expectation Maximization (EM) has also problem like K-means i.e non-convex shape
4. Average Link has not shown good performance
5. Also With the help of NMI score, it is confirmed that the performance of DBSCAN is better

I can conclude on my observations with these datasets that DBSCAN appears to be the better clustering algorithm.

1. https://en.wikipedia.org/wiki/Cluster_analysis (https://en.wikipedia.org/wiki/Cluster_analysis)
2. https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html (https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html)
3. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html)
4. https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html (https://numpy.org/doc/stable/reference/generated/numpy.loadtxt.html)