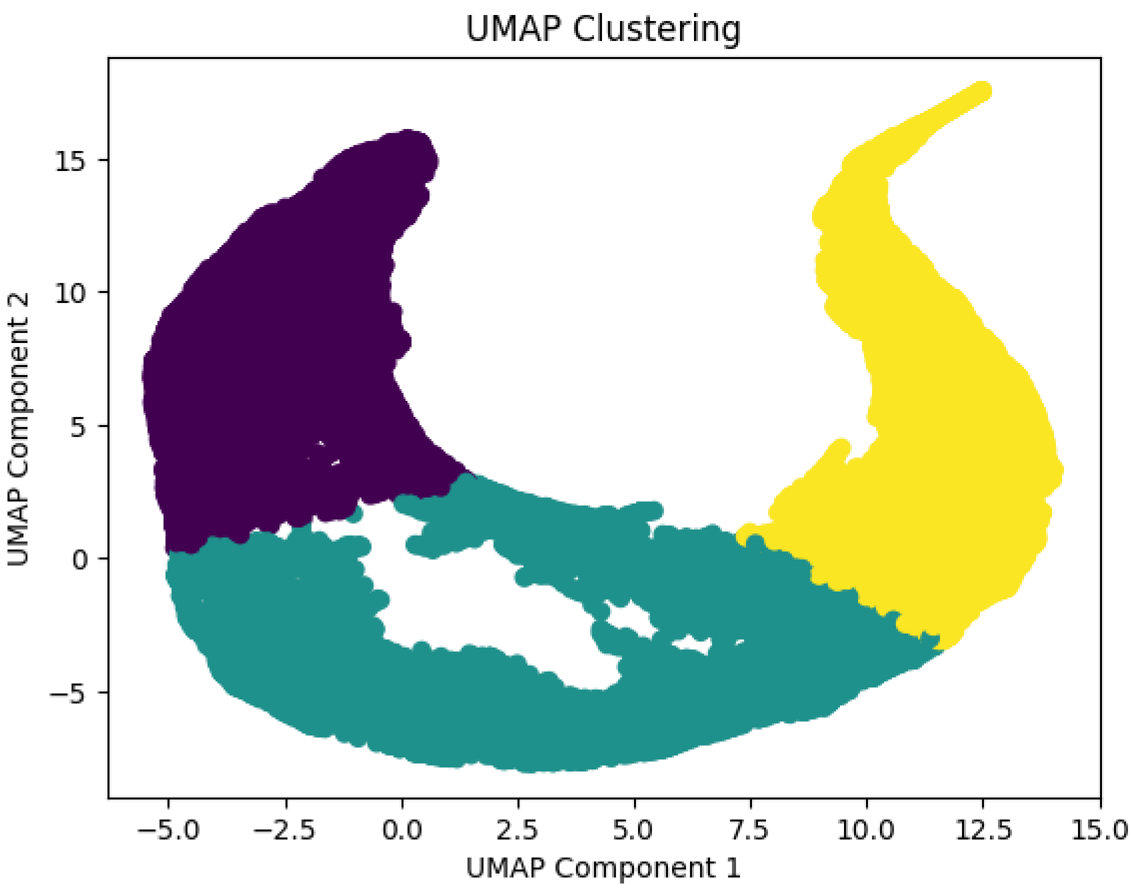
Preparing Data And Preprocessing

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.decomposition import PCA
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler, PowerTransformer
from sklearn.feature_selection import VarianceThreshold
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.neighbors import LocalOutlierFactor
from kneed import KneeLocator
from sklearn.cluster import KMeans
from sklearn.mixture import GaussianMixture
import umap.umap_ as umap

# ---- STEP 1: LOAD DATA ----
df = pd.read_excel('./data_files/Data_re.xlsx')  # Ensure the file exists

# ---- STEP 2: FEATURE SELECTION ----
cols_to_keep = ['g_flux', 'r_flux', 'i_flux', 'y_flux', 'z_flux', 'specz_
df_selected = df[cols_to_keep].copy()

# Compute Flux Color Indices
df_selected['g_r'] = df_selected['g_flux'] - df_selected['r_flux']
df_selected['r_i'] = df_selected['r_flux'] - df_selected['i_flux']
df_selected['i_y'] = df_selected['i_flux'] - df_selected['y_flux']
df_selected['y_z'] = df_selected['y_flux'] - df_selected['z_flux']

# ---- STEP 3: REMOVE OUTLIERS ----
lof = LocalOutlierFactor(n_neighbors=20, contamination=0.02)
outlier_scores = lof.fit_predict(df_selected.drop(columns=['specz_redshif
df_clean = df_selected[outlier_scores == 1].copy()

# ---- STEP 4: SCALING & TRANSFORMING ----
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_clean.drop(columns=['specz_redshift']

# Apply Power Transformation (Yeo-Johnson for normalizing skewed data)
power_transformer = PowerTransformer(method='yeo-johnson')
df_transformed = power_transformer.fit_transform(df_scaled)

# ---- STEP 5: DIMENSIONALITY REDUCTION WITH UMAP ----
reducer = umap.UMAP(n_components=2, n_neighbors=15, min_dist=0.1, random_
X_umap = reducer.fit_transform(df_transformed)

# ---- STEP 6: OPTIONAL PCA FOR EXPLORATION ----
pca = PCA(n_components=3)
X_pca = pca.fit_transform(df_transformed)

# ---- STEP 7: CLUSTERING ----
kmeans = KMeans(n_clusters=3, random_state=42)
df_clean['Cluster'] = kmeans.fit_predict(X_umap)

# ---- PLOT RESULTS ----
```

```
plt.scatter(X_umap[:, 0], X_umap[:, 1], c=df_clean['Cluster'], cmap='viri
plt.xlabel('UMAP Component 1')
plt.ylabel('UMAP Component 2')
plt.title('UMAP Clustering')
plt.show()
```

```
2025-02-21 19:44:36.244662: E external/local_xla/xla/stream_executor/cuda/
cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register
factory for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
E0000 00:00:1740147276.299220    4415 cuda_dnn.cc:8310] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one ha
s already been registered
E0000 00:00:1740147276.314946    4415 cuda_blas.cc:1418] Unable to registe
r cuBLAS factory: Attempting to register factory for plugin cuBLAS when on
e has already been registered
2025-02-21 19:44:36.430661: I tensorflow/core/platform/cpu_feature_guard.c
c:210] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebui
ld TensorFlow with the appropriate compiler flags.
/home/chloy/miniconda3/lib/python3.10/site-packages/sklearn/utils/deprecat
ion.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_f
inite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/home/chloy/miniconda3/lib/python3.10/site-packages/umap/umap_.py:1952: Us
erWarning: n_jobs value 1 overridden to 1 by setting random_state. Use no
seed for parallelism.
  warn(
```



checking

In [2]:
```
print(df.head())
print(df.info())
print(df.describe())
```

```
                                               coord       dec  \
0   b'(179325.3125, 99694.8046875, -21178.96484375)' -5.893433
1  b'(179236.609375, 99349.8203125, -23431.181640... -6.522742
2     b'(179281.5, 99283.5078125, -23368.759765625)' -6.505290
3  b'(179365.171875, 99158.046875, -23259.0839843... -6.474628
4      b'(179366.421875, 99172.25, -23188.84765625)' -6.454993

   g_central_image_pop_10px_rad  g_central_image_pop_15px_rad  \
0                             1                             1
1                             1                             1
2                             1                             1
3                             1                             1
4                             1                             1

   g_central_image_pop_5px_rad  g_cmodel_mag  g_cmodel_magsigma  \
0                            1     20.314907           0.002624
1                            1     22.217360           0.010902
2                            1     21.148739           0.008013
3                            1     18.464205           0.001740
4                            1     20.998287           0.006011

   g_ellipticity  g_half_light_radius  g_isophotal_area  ...  z_minor_axis
\
0          0.147                6.047               603  ...         4.938
1          0.130                3.430                93  ...         2.713
2          0.209                6.597               254  ...         4.351
3          0.525               10.855              1064  ...         4.815
4          0.738                8.261               386  ...         2.279

   z_peak_surface_brightness  z_petro_rad  z_pos_angle  z_sersic_index  \
0                    -8.2933         5.28        36.15           2.193
1                    -7.3657         5.94       -61.78           1.649
2                    -7.6539         9.24        32.76           2.364
3                    -8.5825         6.60        53.15           1.494
4                    -6.8798         5.94        21.16           1.063

         g_flux        r_flux        i_flux        y_flux        z_flux
0  7.482335e-09  2.987891e-08  5.117533e-08  8.117985e-08  6.737572e-08
1  1.297347e-09  5.643225e-09  1.319162e-08  2.141019e-08  1.828079e-08
2  3.471398e-09  1.571304e-08  3.743521e-08  6.504857e-08  5.585498e-08
3  4.114510e-08  1.090914e-07  1.841788e-07  3.080686e-07  2.402212e-07
4  3.987357e-09  1.373544e-08  2.634882e-08  4.449762e-08  3.527128e-08

[5 rows x 89 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40914 entries, 0 to 40913
Data columns (total 89 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   coord                         40914 non-null  object
 1   dec                           40914 non-null  float64
 2   g_central_image_pop_10px_rad  40914 non-null  int64
 3   g_central_image_pop_15px_rad  40914 non-null  int64
 4   g_central_image_pop_5px_rad   40914 non-null  int64
 5   g_cmodel_mag                  40914 non-null  float64
 6   g_cmodel_magsigma             40914 non-null  float64
 7   g_ellipticity                 40914 non-null  float64
 8   g_half_light_radius           40914 non-null  float64
 9   g_isophotal_area              40914 non-null  int64
 10  g_major_axis                  40914 non-null  float64
```

```
 11  g_minor_axis                  40914 non-null  float64
 12  g_peak_surface_brightness     40914 non-null  float64
 13  g_petro_rad                   40914 non-null  float64
 14  g_pos_angle                   40914 non-null  float64
 15  g_sersic_index                40914 non-null  float64
 16  i_central_image_pop_10px_rad  40914 non-null  int64
 17  i_central_image_pop_15px_rad  40914 non-null  int64
 18  i_central_image_pop_5px_rad   40914 non-null  int64
 19  i_cmodel_mag                  40914 non-null  float64
 20  i_cmodel_magsigma             40914 non-null  float64
 21  i_ellipticity                 40914 non-null  float64
 22  i_half_light_radius           40914 non-null  float64
 23  i_isophotal_area              40914 non-null  int64
 24  i_major_axis                  40914 non-null  float64
 25  i_minor_axis                  40914 non-null  float64
 26  i_peak_surface_brightness     40914 non-null  float64
 27  i_petro_rad                   40914 non-null  float64
 28  i_pos_angle                   40914 non-null  float64
 29  i_sersic_index                40914 non-null  float64
 30  object_id                     40914 non-null  int64
 31  r_central_image_pop_10px_rad  40914 non-null  int64
 32  r_central_image_pop_15px_rad  40914 non-null  int64
 33  r_central_image_pop_5px_rad   40914 non-null  int64
 34  r_cmodel_mag                  40914 non-null  float64
 35  r_cmodel_magsigma             40914 non-null  float64
 36  r_ellipticity                 40914 non-null  float64
 37  r_half_light_radius           40914 non-null  float64
 38  r_isophotal_area              40914 non-null  int64
 39  r_major_axis                  40914 non-null  float64
 40  r_minor_axis                  40914 non-null  float64
 41  r_peak_surface_brightness     40914 non-null  float64
 42  r_petro_rad                   40914 non-null  float64
 43  r_pos_angle                   40914 non-null  float64
 44  r_sersic_index                40914 non-null  float64
 45  ra                            40914 non-null  float64
 46  skymap_id                     40914 non-null  int64
 47  specz_dec                     40914 non-null  float64
 48  specz_flag_homogeneous        40914 non-null  bool
 49  specz_mag_i                   40914 non-null  float64
 50  specz_name                    40914 non-null  object
 51  specz_ra                      40914 non-null  float64
 52  specz_redshift                40914 non-null  float64
 53  specz_redshift_err            40914 non-null  float64
 54  x_coord                       40914 non-null  float64
 55  y_central_image_pop_10px_rad  40914 non-null  int64
 56  y_central_image_pop_15px_rad  40914 non-null  int64
 57  y_central_image_pop_5px_rad   40914 non-null  int64
 58  y_cmodel_mag                  40914 non-null  float64
 59  y_cmodel_magsigma             40914 non-null  float64
 60  y_coord                       40914 non-null  float64
 61  y_ellipticity                 40914 non-null  float64
 62  y_half_light_radius           40914 non-null  float64
 63  y_isophotal_area              40914 non-null  int64
 64  y_major_axis                  40914 non-null  float64
 65  y_minor_axis                  40914 non-null  float64
 66  y_peak_surface_brightness     40914 non-null  float64
 67  y_petro_rad                   40914 non-null  float64
 68  y_pos_angle                   40914 non-null  float64
 69  y_sersic_index                40914 non-null  float64
 70  z_central_image_pop_10px_rad  40914 non-null  int64
```

```
 71  z_central_image_pop_15px_rad  40914 non-null  int64
 72  z_central_image_pop_5px_rad   40914 non-null  int64
 73  z_cmodel_mag                  40914 non-null  float64
 74  z_cmodel_magsigma             40914 non-null  float64
 75  z_ellipticity                 40914 non-null  float64
 76  z_half_light_radius           40914 non-null  float64
 77  z_isophotal_area              40914 non-null  int64
 78  z_major_axis                  40914 non-null  float64
 79  z_minor_axis                  40914 non-null  float64
 80  z_peak_surface_brightness     40914 non-null  float64
 81  z_petro_rad                   40914 non-null  float64
 82  z_pos_angle                   40914 non-null  float64
 83  z_sersic_index                40914 non-null  float64
 84  g_flux                        40914 non-null  float64
 85  r_flux                        40914 non-null  float64
 86  i_flux                        40914 non-null  float64
 87  y_flux                        40914 non-null  float64
 88  z_flux                        40914 non-null  float64
dtypes: bool(1), float64(64), int64(22), object(2)
memory usage: 27.5+ MB
None
                dec  g_central_image_pop_10px_rad  \
count  40914.000000                  40914.000000
mean       4.772650                      1.016278
std       14.789896                      0.202830
min       -7.217183                      0.000000
25%       -0.960840                      1.000000
50%        0.311508                      1.000000
75%        1.712535                      1.000000
max       53.260936                      3.000000

       g_central_image_pop_15px_rad  g_central_image_pop_5px_rad  \
count                  40914.000000                 40914.000000
mean                       1.032141                     0.998142
std                        0.240109                     0.160037
min                        0.000000                     0.000000
25%                        1.000000                     1.000000
50%                        1.000000                     1.000000
75%                        1.000000                     1.000000
max                        4.000000                     3.000000

        g_cmodel_mag  g_cmodel_magsigma  g_ellipticity  g_half_light_radius
\
count  40914.000000       40914.000000   40914.000000         40914.000000
mean      21.260192           0.010414       0.230579             6.509069
std        1.882030           0.091981       0.171495             3.625851
min       14.753462           0.000149       0.000000             0.000000
25%       19.733286           0.001531       0.095000             3.905000
50%       21.492703           0.004890       0.190000             5.679000
75%       22.614446           0.011530       0.328000             7.976000
max       30.276308          17.593214       0.896000            26.335000

       g_isophotal_area  g_major_axis  ...  z_minor_axis  \
count      40914.000000  40914.000000  ...  40914.000000
mean         694.781786      5.523996  ...      4.064275
std          905.508200      4.095328  ...      2.145741
min            0.000000      0.000000  ...      0.000000
25%          122.000000      2.706250  ...      2.308250
50%          262.000000      3.938000  ...      3.740000
75%          947.000000      7.249000  ...      5.250000
```

```
max        6434.000000    33.320000  ...    16.530000

       z_peak_surface_brightness   z_petro_rad   z_pos_angle  z_sersic_ind
ex  \
count                40914.000000  40914.000000  40914.000000    40914.0000
00
mean                    -6.819224      6.249483      1.738910        1.6644
49
std                      1.418684      1.392911     52.237031        0.8703
28
min                    -10.477100      0.000000    -89.990000        0.0000
00
25%                     -7.830100      5.280000    -43.900000        1.0370
00
50%                     -7.000600      5.940000      3.525000        1.5230
00
75%                     -5.935925      6.600000     46.870000        2.0950
00
max                      0.000000     10.560000     90.000000        9.9740
00

             g_flux        r_flux        i_flux        y_flux         z_flu
x
count  4.091400e+04  4.091400e+04  4.091400e+04  4.091400e+04  4.091400e+0
4
mean   1.210340e-08  2.768307e-08  4.473075e-08  6.989669e-08  5.870488e-0
8
std    2.736938e-08  5.694149e-08  8.773860e-08  1.366205e-07  1.134256e-0
7
min    7.753125e-13  1.954729e-12  8.507724e-12  3.916208e-11  3.854886e-1
1
25%    8.999569e-10  2.390220e-09  4.067133e-09  6.001859e-09  5.229988e-0
9
50%    2.528826e-09  7.571469e-09  1.592592e-08  2.686640e-08  2.237941e-0
8
75%    1.278456e-08  3.078038e-08  4.926195e-08  7.639837e-08  6.444087e-0
8
max    1.254918e-06  2.506786e-06  3.731277e-06  5.268088e-06  4.333962e-0
6

[8 rows x 86 columns]
```

Clustering (DBScan, K means and Gaussian Mixture Method)

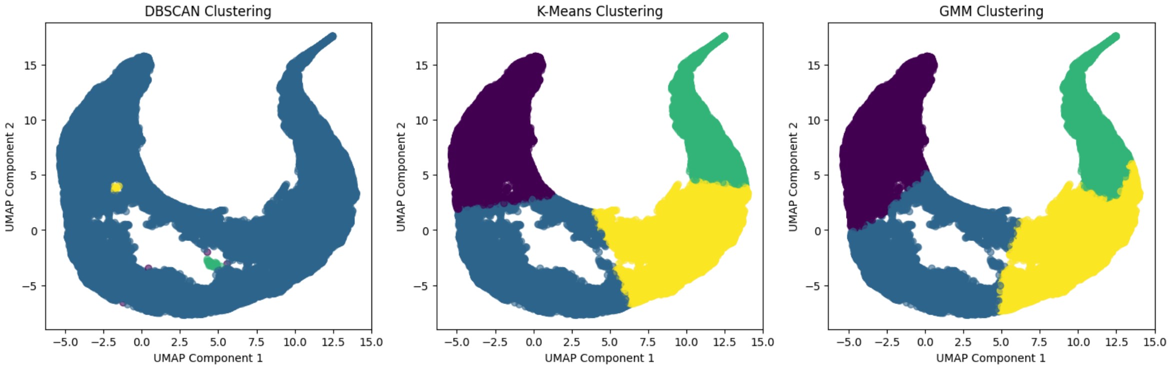
In [3]:

```python
from sklearn.cluster import DBSCAN, KMeans
from sklearn.mixture import GaussianMixture
from sklearn.neighbors import NearestNeighbors
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np
from kneed import KneeLocator

# ---- STEP 6: DETERMINE OPTIMAL DBSCAN EPS ----
k = 5  # Typically, min_samples value
nearest_neighbors = NearestNeighbors(n_neighbors=k)
nearest_neighbors.fit(X_umap)
distances, indices = nearest_neighbors.kneighbors(X_umap)

# Sort distances to find the "knee" point
distances = np.sort(distances[:, -1])
```

```
# Use KneeLocator to find optimal epsilon
knee_locator = KneeLocator(range(len(distances)), distances, curve="conve
optimal_eps = distances[knee_locator.elbow]
print(f"Optimal eps for DBSCAN: {optimal_eps:.3f}")

# ---- STEP 6A: APPLY DBSCAN CLUSTERING WITH OPTIMAL EPS ----
dbscan = DBSCAN(eps=optimal_eps, min_samples=k, metric='euclidean')
cluster_labels_dbscan = dbscan.fit_predict(X_umap)
df_clean.loc[:, 'cluster_dbscan'] = cluster_labels_dbscan

# ---- STEP 6B: APPLY K-MEANS & GMM ----
inertia = []
silhouette_scores = []
k_range = range(2, 10)

for k in k_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    labels = kmeans.fit_predict(X_umap)
    inertia.append(kmeans.inertia_)
    score = silhouette_score(X_umap, labels)
    silhouette_scores.append(score)

knee_locator = KneeLocator(k_range, inertia, curve="convex", direction="d
optimal_k = knee_locator.elbow
print(f"Optimal k for K-Means/GMM: {optimal_k}")

# Apply K-Means with the optimal k
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init='auto')
df_clean.loc[:, 'cluster_kmeans'] = kmeans.fit_predict(X_umap)

# Apply Gaussian Mixture Model (GMM)
gmm = GaussianMixture(n_components=optimal_k, random_state=42)
df_clean.loc[:, 'cluster_gmm'] = gmm.fit_predict(X_umap)

# ---- STEP 7: COMPUTE SILHOUETTE SCORES ----
silhouette_kmeans = silhouette_score(X_umap, df_clean['cluster_kmeans'])
silhouette_gmm = silhouette_score(X_umap, df_clean['cluster_gmm'])
silhouette_dbscan = silhouette_score(X_umap[df_clean['cluster_dbscan'] !=
                                     df_clean['cluster_dbscan'][df_clean[

print(f"Silhouette Scores - KMeans: {silhouette_kmeans:.3f}, GMM: {silhou

# ---- STEP 8: VISUALIZE CLUSTERING METHODS (2D PLOTS) ----
fig, ax = plt.subplots(1, 3, figsize=(18, 5))

# DBSCAN 2D Plot
ax[0].scatter(X_umap[:, 0], X_umap[:, 1], c=cluster_labels_dbscan, cmap='
ax[0].set_title("DBSCAN Clustering")
ax[0].set_xlabel("UMAP Component 1")
ax[0].set_ylabel("UMAP Component 2")

# K-Means 2D Plot
ax[1].scatter(X_umap[:, 0], X_umap[:, 1], c=df_clean['cluster_kmeans'], c
ax[1].set_title("K-Means Clustering")
ax[1].set_xlabel("UMAP Component 1")
ax[1].set_ylabel("UMAP Component 2")

# GMM 2D Plot
ax[2].scatter(X_umap[:, 0], X_umap[:, 1], c=df_clean['cluster_gmm'], cmap
```

```
ax[2].set_title("GMM Clustering")
ax[2].set_xlabel("UMAP Component 1")
ax[2].set_ylabel("UMAP Component 2")

plt.show()
```

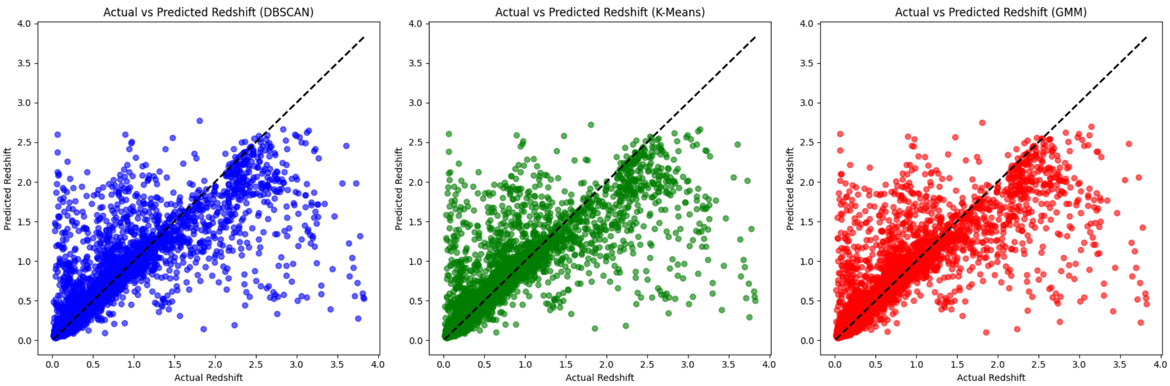
```
Optimal eps for DBSCAN: 0.323
Optimal k for K-Means/GMM: 4
Silhouette Scores - KMeans: 0.486, GMM: 0.455, DBSCAN: -0.422
```



Random Forest

In [4]:
```
# ---- STEP 8: RANDOM FOREST REGRESSION ----
# Train separate Random Forest models for each clustering method using fl

results = {}

for cluster_type in ['cluster_dbscan', 'cluster_kmeans', 'cluster_gmm']:
    df_temp = pd.get_dummies(df_clean, columns=[cluster_type], prefix=[f'
    X = df_temp[['g_flux', 'r_flux', 'i_flux', 'y_flux', 'z_flux'] +
                [col for col in df_temp.columns if col.startswith(f'clust
    y = df_temp['specz_redshift']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X_train, y_train)

    y_pred = rf.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    results[cluster_type] = {'MSE': mse, 'R2': r2, 'y_pred': y_pred}

# ---- STEP 9: VISUALIZE REGRESSION RESULTS ----
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
titles = ['DBSCAN', 'K-Means', 'GMM']
colors = ['blue', 'green', 'red']

for i, cluster_type in enumerate(results.keys()):
    ax[i].scatter(y_test, results[cluster_type]['y_pred'], alpha=0.6, col
    ax[i].set_xlabel('Actual Redshift')
    ax[i].set_ylabel('Predicted Redshift')
    ax[i].set_title(f'Actual vs Predicted Redshift ({titles[i]})')
    ax[i].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()]

plt.tight_layout()
plt.show()
```

```python
# ---- STEP 10: COMPARE RESULTS ACROSS CLUSTERING METHODS ----
print("Random Forest Regression Results:")
for cluster_type, metrics in results.items():
    print(f"{cluster_type.upper()} - MSE: {metrics['MSE']:.6f}, R2: {metr
```




```
Random Forest Regression Results:
CLUSTER_DBSCAN - MSE: 0.103494, R2: 0.673538
CLUSTER_KMEANS - MSE: 0.103177, R2: 0.674537
CLUSTER_GMM - MSE: 0.103596, R2: 0.673214
```

Checking

In [5]:

```python
print(df_clean.head())
print(df_clean.columns)
```

```
        g_flux        r_flux        i_flux        y_flux        z_flux  \
0  7.482335e-09  2.987891e-08  5.117533e-08  8.117985e-08  6.737572e-08
1  1.297347e-09  5.643225e-09  1.319162e-08  2.141019e-08  1.828079e-08
2  3.471398e-09  1.571304e-08  3.743521e-08  6.504857e-08  5.585498e-08
3  4.114510e-08  1.090914e-07  1.841788e-07  3.080686e-07  2.402212e-07
4  3.987357e-09  1.373544e-08  2.634882e-08  4.449762e-08  3.527128e-08

   specz_redshift           g_r           r_i           i_y           y_z
\
0         0.31652 -2.239657e-08 -2.129643e-08 -3.000452e-08  1.380413e-08
1         0.56769 -4.345878e-09 -7.548391e-09 -8.218576e-09  3.129404e-09
2         0.53428 -1.224164e-08 -2.172218e-08 -2.761336e-08  9.193588e-09
3         0.11878 -6.794635e-08 -7.508739e-08 -1.238898e-07  6.784739e-08
4         0.23497 -9.748082e-09 -1.261338e-08 -1.814880e-08  9.226337e-09

   Cluster  cluster_dbscan  cluster_kmeans  cluster_gmm
0        2               0               3            3
1        1               0               1            1
2        1               0               3            3
3        2               0               2            2
4        1               0               3            3
Index(['g_flux', 'r_flux', 'i_flux', 'y_flux', 'z_flux', 'specz_redshift',
       'g_r', 'r_i', 'i_y', 'y_z', 'Cluster', 'cluster_dbscan',
       'cluster_kmeans', 'cluster_gmm'],
      dtype='object')
```

SVR and UMAP

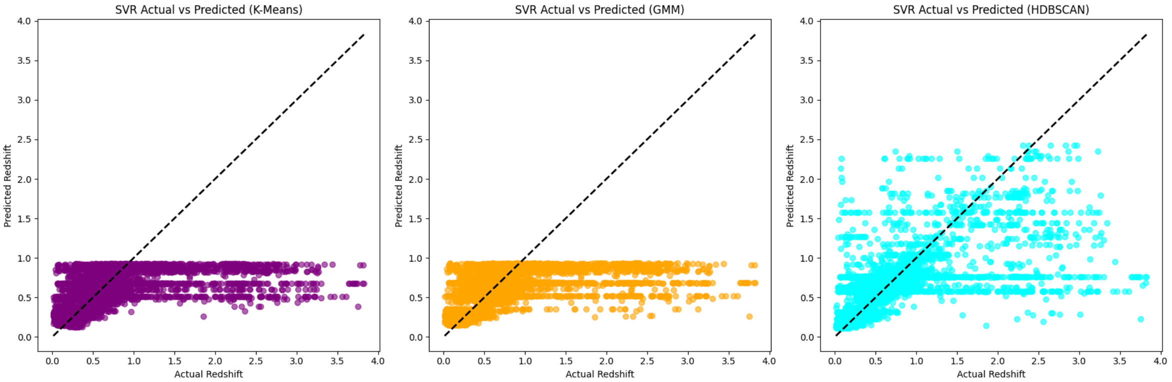
In [6]:

```python
# Add-Ons for Photometric Redshift Estimation
# Fixing UMAP Import Issue and Enhancing Preprocessing

# ---- STEP 11: Install and Import UMAP Properly ----
```

```
# Ensure proper UMAP installation: pip install umap-learn
from umap import UMAP

umap = UMAP(n_neighbors=15, min_dist=0.1, n_components=2, random_state=42
X_umap = umap.fit_transform(df_transformed)

# ---- STEP 12: Add More Color Indices (Using Flux Values) ----
df_clean['u_g'] = df_clean['g_flux'] - df_clean['r_flux']
df_clean['g_r'] = df_clean['r_flux'] - df_clean['i_flux']
df_clean['r_i'] = df_clean['i_flux'] - df_clean['y_flux']
df_clean['i_z'] = df_clean['y_flux'] - df_clean['z_flux']

# ---- STEP 13: Additional Clustering (HDBSCAN) for SVR Comparison ----
try:
    from hdbscan import HDBSCAN
except ImportError:
    print("HDBSCAN not installed. Use: pip install hdbscan")
    raise

hdbscan = HDBSCAN(min_cluster_size=8)
df_clean['cluster_hdbscan'] = hdbscan.fit_predict(X_umap)

# ---- STEP 14: SVR Regression for Redshift Estimation ----
from sklearn.svm import SVR
svr_results = {}

for cluster_type in ['cluster_kmeans', 'cluster_gmm', 'cluster_hdbscan']:
    df_temp = pd.get_dummies(df_clean, columns=[cluster_type], prefix=[f'
    X = df_temp.drop(columns=['specz_redshift'])
    y = df_temp['specz_redshift']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0

    svr = SVR(kernel='rbf', C=1.0, epsilon=0.1)
    svr.fit(X_train, y_train)
    y_pred = svr.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    svr_results[cluster_type] = {'MSE': mse, 'R2': r2, 'y_pred': y_pred}

# ---- STEP 15: Plot SVR Results ----
fig, ax = plt.subplots(1, 3, figsize=(18, 6))
titles = ['K-Means', 'GMM', 'HDBSCAN']
colors = ['purple', 'orange', 'cyan']

for i, cluster_type in enumerate(svr_results.keys()):
    ax[i].scatter(y_test, svr_results[cluster_type]['y_pred'], alpha=0.6,
    ax[i].set_title(f'SVR Actual vs Predicted ({titles[i]})')
    ax[i].set_xlabel('Actual Redshift')
    ax[i].set_ylabel('Predicted Redshift')
    ax[i].plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()]

plt.tight_layout()
plt.show()

# ---- Print SVR Performance Metrics ----
print('SVR Regression Results:')
```

```
for cluster_type, metrics in svr_results.items():
    print(f"{cluster_type.upper()} - MSE: {metrics['MSE']:.6f}, R2: {metr
```

```
/home/chloy/miniconda3/lib/python3.10/site-packages/sklearn/utils/deprecat
ion.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_f
inite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/home/chloy/miniconda3/lib/python3.10/site-packages/umap/umap_.py:1952: Us
erWarning: n_jobs value 1 overridden to 1 by setting random_state. Use no
seed for parallelism.
  warn(
/home/chloy/miniconda3/lib/python3.10/site-packages/sklearn/utils/deprecat
ion.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_f
inite' in 1.6 and will be removed in 1.8.
  warnings.warn(
/home/chloy/miniconda3/lib/python3.10/site-packages/sklearn/utils/deprecat
ion.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_f
inite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```




```
SVR Regression Results:
CLUSTER_KMEANS - MSE: 0.227899, R2: 0.281111
CLUSTER_GMM - MSE: 0.226127, R2: 0.286701
CLUSTER_HDBSCAN - MSE: 0.167378, R2: 0.472019
```