Mô Tả

1. Khai báo thư viện :

In [1]:
```
import pandas as pd
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
```

2. Đọc dữ liệu

Dữ liệu được đọc từ tệp CSV chứa thông tin về giá SMP và SMPcap trong năm 2021. Đường dẫn tới tệp CSV được chỉ định và dữ liệu được tải vào một DataFrame của pandas.

In [2]:
```
path = "/kaggle/input/giasmp/GiaSMP.csv"
data = pd.read_csv(path, encoding="latin-1",sep=";")
data
msv = 2151260829
col1 = msv%15 + 1
col2 = msv%15 + 2
data_3 = data[[str(col1),str(col2)]]
data_3.head(5)
```

Out[2]:

3. Mô Tả Dữ Liệu

In [4]:
```
# Lấy dữ liệu từ cột được chọn
selected_data = data.iloc[:, [10, 11]]
```

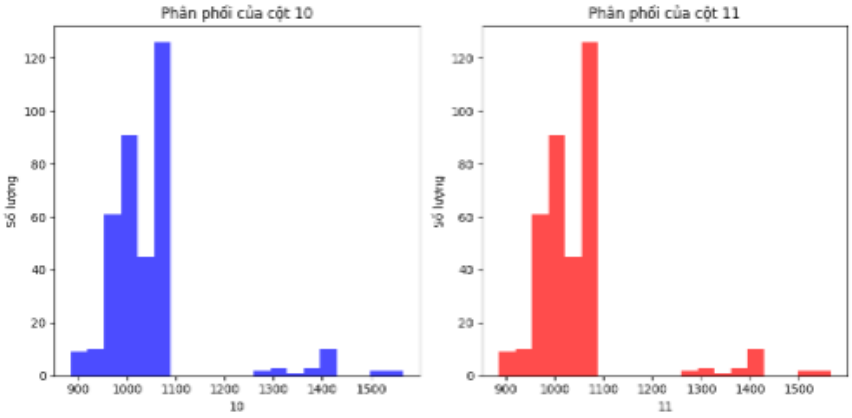
In [5]:
```
#Mo ta du lieu
print("Thông tin tổng quan về dữ liệu:")
data_3.info()
```

```
Thông tin tổng quan về dữ liệu:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 365 entries, 0 to 364
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   10      365 non-null    float64
 1   11      365 non-null    float64
dtypes: float64(2)
```

4. Trực Quan Hóa Dữ Liệu

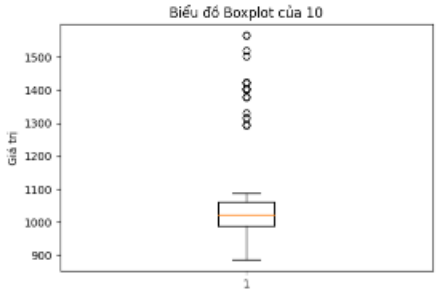
Các biểu đồ histogram, boxplot, line plot và violin plot được tạo ra để trực quan hóa sự phân bố của các biến số trong dữ liệu.

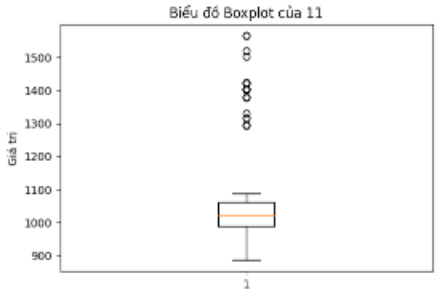
In [8]:
```
# Kiểm tra giá trị thiếu trong dữ liệu
plt.figure(figsize=(18, 5))
plt.subplot(1, 2, 1)
plt.hist(data_3[str(col1)], bins=20, color='blue', alpha=0.7)
plt.title(f'Phân phối của cột {str(col1)}')
plt.xlabel(str(col1))
plt.ylabel('Số lượng')

plt.subplot(1, 2, 2)
plt.hist(data_3[str(col2)], bins=20, color='red', alpha=0.7)
plt.title(f'Phân phối của cột {str(col2)}')
plt.xlabel(str(col2))
plt.ylabel('Số lượng')

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



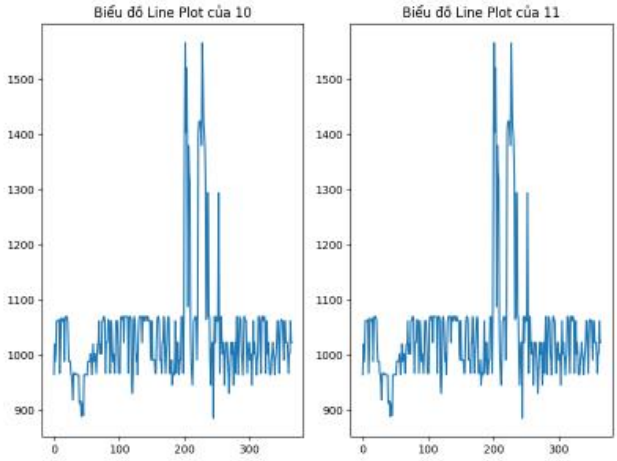
In [9]:

```
# Biểu đồ boxplot cho phân phối và giá trị ngoại lệ của cột col1
plt.figure(figsize=(6, 4))
plt.boxplot(data_34[str(col1)])
plt.title(f'Biểu đồ Boxplot của {col1}')
plt.ylabel('Giá trị')
plt.show()
# Biểu đồ boxplot cho phân phối và giá trị ngoại lệ của cột col2
plt.figure(figsize=(6, 4))
plt.boxplot(data_34[str(col2)])
plt.title(f'Biểu đồ Boxplot của {col2}')
plt.ylabel('Giá trị')
plt.show()
```

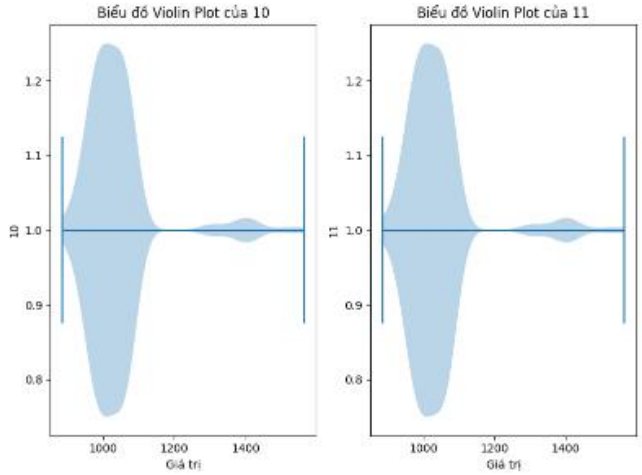




In [10]:
```
# Biểu đồ Line Plot cho col1 và col2
plt.figure(figsize=(8, 6))
plt.subplot(1, 2, 1)
plt.plot(data_3[str(col1)])
plt.title(f'Biểu đồ Line Plot của {str(col1)}')

plt.subplot(1, 2, 2)
plt.plot(data_3[str(col2)])
plt.title(f'Biểu đồ Line Plot của {str(col2)}')

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



In [11]:
```
# Biểu đồ Violin Plot cho col1 và col2
plt.figure(figsize=(8, 6))
plt.subplot(1, 2, 1)
plt.violinplot(data_3[str(col1)], vert=False)
plt.title(f'Biểu đồ Violin Plot của {str(col1)}')
plt.xlabel('Giá trị')
plt.ylabel(str(col1))

plt.subplot(1, 2, 2)
plt.violinplot(data_3[str(col2)], vert=False)
plt.title(f'Biểu đồ Violin Plot của {str(col2)}')
plt.xlabel('Giá trị')
plt.ylabel(str(col2))

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

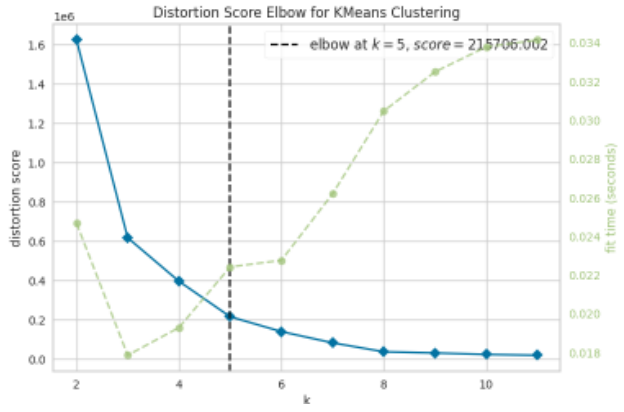


5. Tách Dữ Liệu

Dữ liệu được tách bằng các phương pháp Gaussian Mixture Model (GMM), KMeans, và DBSCAN. Số cụm tối ưu cho KMeans được xác định bằng phương pháp Elbow.

In [69]:
```
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans

print('Tìm cụm tối ưu Kmeans')

# Assuming 'selected_data' is already a 2D array-like structure (e.g., a DataFrame or a numpy array)
# If 'selected_data' is 1D, you can convert it to 2D using selected_data = selected_data.reshape(-1, 1)
selected_data_2d = selected_data  # No reshape needed if already 2D

# Initialize the KMeans model
kmeans = KMeans(random_state=23, n_init=10)

# Initialize the KElbowVisualizer with the KMeans model and specify the range of clusters to search
elbow_visualizer = KElbowVisualizer(kmeans, k=(2, 12))

# Fit the visualizer to the data
elbow_visualizer.fit(selected_data_2d)

# Display the elbow plot
elbow_visualizer.show()
```

```
Tìm cụm tối ưu Kmeans
```



6. Trực Quan Hóa Kết Quả Phân Cụm

Biểu đồ phân cụm được tạo ra để trực quan hóa kết quả phân cụm của các phương pháp GMM, KMeans và DBSCAN.

In [70]:

```
# Thực hiện Gaussian Mixture Model
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans


k = 5  # or another appropriate number based on your previous analysis


gmm = GaussianMixture(n_components=k, random_state=23)  # n_components can be adjusted
gmm.fit(selected_data.values)
gmm_labels = gmm.predict(selected_data.values)

print("GMM Labels:")
print(gmm_labels)

kmeans = KMeans(n_clusters=k, n_init=10, random_state=23)  # n_clusters can be adjusted
kmeans.fit(selected_data.values)
kmeans_labels = kmeans.predict(selected_data.values)

print("KMeans Labels:")
print(kmeans_labels)
```

```
GMM Labels:
[2 2 2 3 0 0 0 0 0 2 2 0 0 0 0 0 2 3 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2
 0 0 0 0 0 2 2 2 2 0 0 0 2 2 0 2 2 2 2 2 2 2 0 0 0 2 2 2 2 0 0 0 0 0 2 0 0 0
 0 0 0 2 0 0 0 0 0 2 2 2 0 0 0 2 2 2 2 2 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0
 2 2 0 2 2 2 2 2 2 2 2 0 0 0 0 2 2 2 2 2 2 2 0 0 2 2 2 2 2 0 0 0 2 2 2 2 2 2
 2 0 2 2 2 2 2 2 2 2 2 2 0 0 0 2 0 1 1 1 1 1 2 1 1 4 0 2 2 2 2 0 0 0 0 0 2 0 1
 1 1 1 1 1 1 1 1 1 1 4 0 0 4 4 0 0 2 2 2 2 2 2 2 2 0 2 0 0 0 0 4 0 2 0 0 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 0 0 2 2 2 2 0 0 0 0 2
 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 0 0 0 2 2 0 0 0 0 0 0 2 2 0 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 0 0 2 2 2 2 0 0 0 0 2 2 2 0 0 2 2 2 2 2 2 2 2 0 2 2]
KMeans Labels:
[2 4 4 4 0 0 0 0 0 2 4 0 0 0 0 0 4 0 0 0 0 0 4 4 4 4 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 4 2 4 4 4 4 4 4 4 2 4 4 4 0 4 4 4 4
 0 0 0 0 0 2 4 4 4 0 0 0 2 4 0 4 4 4 4 2 4 0 0 0 4 2 2 2 0 0 0 0 0 4 0 0 0
 0 0 0 2 0 0 0 0 2 2 2 0 0 0 4 4 4 2 4 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0
 4 4 0 4 4 4 4 2 2 4 0 0 0 0 4 4 4 2 4 4 0 0 4 2 4 4 0 0 0 2 2 2 4 2 2 2 2
 4 0 2 4 4 4 2 2 4 4 0 0 0 2 0 3 1 3 1 3 0 3 3 3 0 4 2 2 4 0 0 0 0 0 4 0 3
 3 3 3 3 3 1 1 3 3 3 3 0 0 3 3 0 0 4 2 4 4 4 2 4 0 4 0 0 0 0 3 0 2 0 0 4 4
 4 4 2 4 4 4 4 4 2 2 2 4 4 4 4 4 2 4 4 2 4 0 4 2 4 0 0 0 4 4 2 4 0 0 0 0 4
 2 4 4 4 4 4 4 2 4 0 4 4 4 4 2 4 4 0 0 0 4 2 0 0 0 0 0 0 2 4 0 4 4 4 2 2 2
 4 4 4 4 4 2 4 4 4 0 0 4 2 4 0 0 0 0 4 4 0 0 4 4 4 4 2 4 4 0 4 4]
```

In [71]:

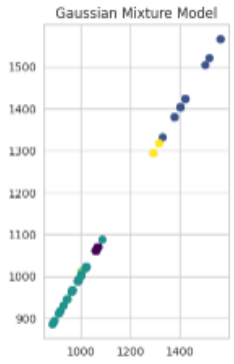
```
from sklearn.mixture import GaussianMixture
from sklearn.cluster import KMeans, DBSCAN


k = 5  # or another appropriate number based on your previous analysis

# Gaussian
gmm = GaussianMixture(n_components=k, random_state=23)  # n_components can be adjusted
gmm.fit(selected_data.values)
gmm_labels = gmm.predict(selected_data.values)

# KMeans
kmeans = KMeans(n_clusters=k, n_init=10, random_state=23)  # n_clusters can be adjusted
kmeans.fit(selected_data.values)
kmeans_labels = kmeans.predict(selected_data.values)

# DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)  # eps and min_samples can be adjusted
dbscan_labels = dbscan.fit_predict(selected_data.values)

print("Gaussian Mixture Model labels:")
print(gmm_labels)
print("\nKMeans labels:")
print(kmeans_labels)
print("\nDBSCAN labels:")
print(dbscan_labels)
```

```
Gaussian Mixture Model labels:
[2 2 2 3 0 0 0 0 0 2 2 0 0 0 0 0 2 3 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2
 0 0 0 0 0 2 2 2 2 0 0 0 2 2 0 2 2 2 2 2 2 0 0 0 2 2 2 2 2 0 0 0 0 0 2 0 0 0
 0 0 0 2 0 0 0 0 2 2 2 0 0 0 2 2 2 2 2 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0
 2 2 0 2 2 2 2 2 2 2 2 0 0 0 0 2 2 2 2 2 2 0 0 2 2 2 2 2 0 0 0 2 2 2 2 2 2 2 2
 2 0 2 2 2 2 2 2 2 2 2 0 0 0 2 0 1 1 1 1 1 2 1 1 4 0 2 2 2 2 0 0 0 0 0 2 0 1
 1 1 1 1 1 1 1 1 1 1 4 0 0 4 4 0 0 2 2 2 2 2 2 2 0 2 0 0 0 0 4 0 2 0 0 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 2 2 0 0 0 2 2 2 2 0 0 0 0 2
 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 0 0 2 2 0 0 0 0 0 0 2 2 0 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 0 0 2 2 2 0 0 0 0 2 2 0 0 2 2 2 2 2 2 2 2 0 2 2]

KMeans labels:
[2 4 4 4 0 0 0 0 0 2 4 0 0 0 0 0 4 0 0 0 0 0 4 4 4 4 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 4 4 4 4 4 2 4 4 4 4 4 4 4 2 4 4 4 4 4 4
 0 0 0 0 0 2 4 4 4 0 0 0 2 4 0 4 4 4 4 2 4 0 0 0 4 2 2 2 0 0 0 0 0 4 0 0 0
 0 0 0 2 0 0 0 0 2 2 2 0 0 0 4 4 4 2 4 0 0 0 0 0 4 0 0 0 0 0 0 0 0 0 0 0 0
 4 4 0 4 4 4 4 4 2 2 4 0 0 0 0 4 4 4 2 4 4 0 0 4 2 4 4 4 0 0 4 2 2 2 4 2 2 2 2
 4 0 2 4 4 4 2 2 4 4 0 0 0 2 0 3 1 3 1 3 0 3 3 3 3 0 4 2 2 4 0 0 0 0 0 4 0 3
 3 3 3 3 3 1 1 3 3 3 3 0 0 3 3 0 0 4 2 4 4 4 2 4 0 4 0 0 0 0 3 0 2 0 0 4 4
 4 4 2 4 4 4 4 4 4 2 2 2 4 4 4 4 4 2 4 4 2 4 0 4 2 4 0 0 0 4 4 2 4 0 0 0 0 4
 2 4 4 4 4 4 4 4 2 4 0 4 4 4 4 4 2 4 4 0 0 0 4 2 0 0 0 0 0 0 2 4 0 4 4 4 2 2 2
 4 4 4 4 4 2 4 4 4 0 0 4 2 4 0 0 0 0 4 4 0 0 4 4 4 4 4 2 4 4 0 4 4]

DBSCAN labels:
[ 0  1  2 -1  3  3  3  3  4  5  6 -1  3  3  3 -1  2 -1  7  7  7  3  1  2
  2  2  5  8  9 -1  5  0  5  5  0  0  8  8  8 -1 -1 -1 -1 -1 -1 -1  8  8
  0  0  0  8  2  2  2  2  6  2  0  2  1  2  6  6  2  5  6  1  1  3  6  6
  1  6  3  3  7  7  3  5 10 10 10  4  3  3  5  6  3 10  2  6  2  5  6  3
  3  3 10  5  5  5  3  7  4  7  7 10  7  7  7  7  7  7  5  3  7  7  4  9
 -1  5  3  7  3 10  2  6  0 10  3  7  7  7  7 10  7  7  3  7  7  7  3  7
  7  3  3  3  6 11  3 11 11 11 11  5  5 10  3  7  7  3 11 11 11  5 10 10
  7  4 11  5  6 10  3  7  4  5  5  5 11  9  0  5  8 11  3  5 11 10 10  0
  5 11 11  7  7  3  5  7 12 -1 12 -1 12 -1 -1 -1 -1  3 11  0  9 11  3  4
  7  7  3 11  7 12 -1 -1 -1 12 -1 -1 -1 -1 12 -1 -1  4  7 -1 -1  7  7 11
  9  6 10 10 -1 10  7 10  7  3  3  3 -1  7  5  3  7 10  6 10  6  9 11 10
 10 10 11  5 -1  0  6 10  6 10 11  9  6 10  5 10  7 10  0 10  7  3  4  6
 11  8  6  7  7  3  3  6  5  6 10 10 11  6 11  9  6  3  6 10  6 11  8 11
  6  7  7  3  6  5  7  7  7  3  7  3  5 10  3  6 10  6  5  8  5 11  6 10
  6 11  0 11  6 10  3  3 10  5 10  3  4  3  3 10 11  3  3 10 10 10  6  5
  6  6  3 10 10]
```

In [72]:

In [73]:
```
plt.figure(figsize=(18, 5))
plt.subplot(1, 3, 1)
plt.scatter(selected_data.iloc[:, 0], selected_data.iloc[:, 1], c=gmm_labels, cmap='viridis')
plt.title('Gaussian Mixture Model')
```

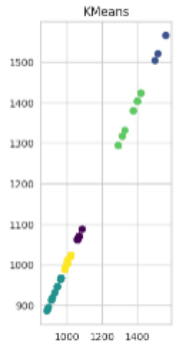
Out[73]:
```
Text(0.5, 1.0, 'Gaussian Mixture Model')
```



In [74]:

In [74]:
```
plt.subplot(1, 3, 2)
plt.scatter(selected_data.iloc[:, 0], selected_data.iloc[:, 1], c=kmeans_labels, cmap='viridis')
plt.title('KMeans')
```

Out[74]:
```
Text(0.5, 1.0, 'KMeans')
```



In [75]:

```
plt.subplot(1, 3, 3)
plt.scatter(selected_data.iloc[:, 0], selected_data.iloc[:, 1], c=dbscan_labels, cmap='viridis')
plt.title('DBSCAN')
plt.tight_layout()
plt.show()
```