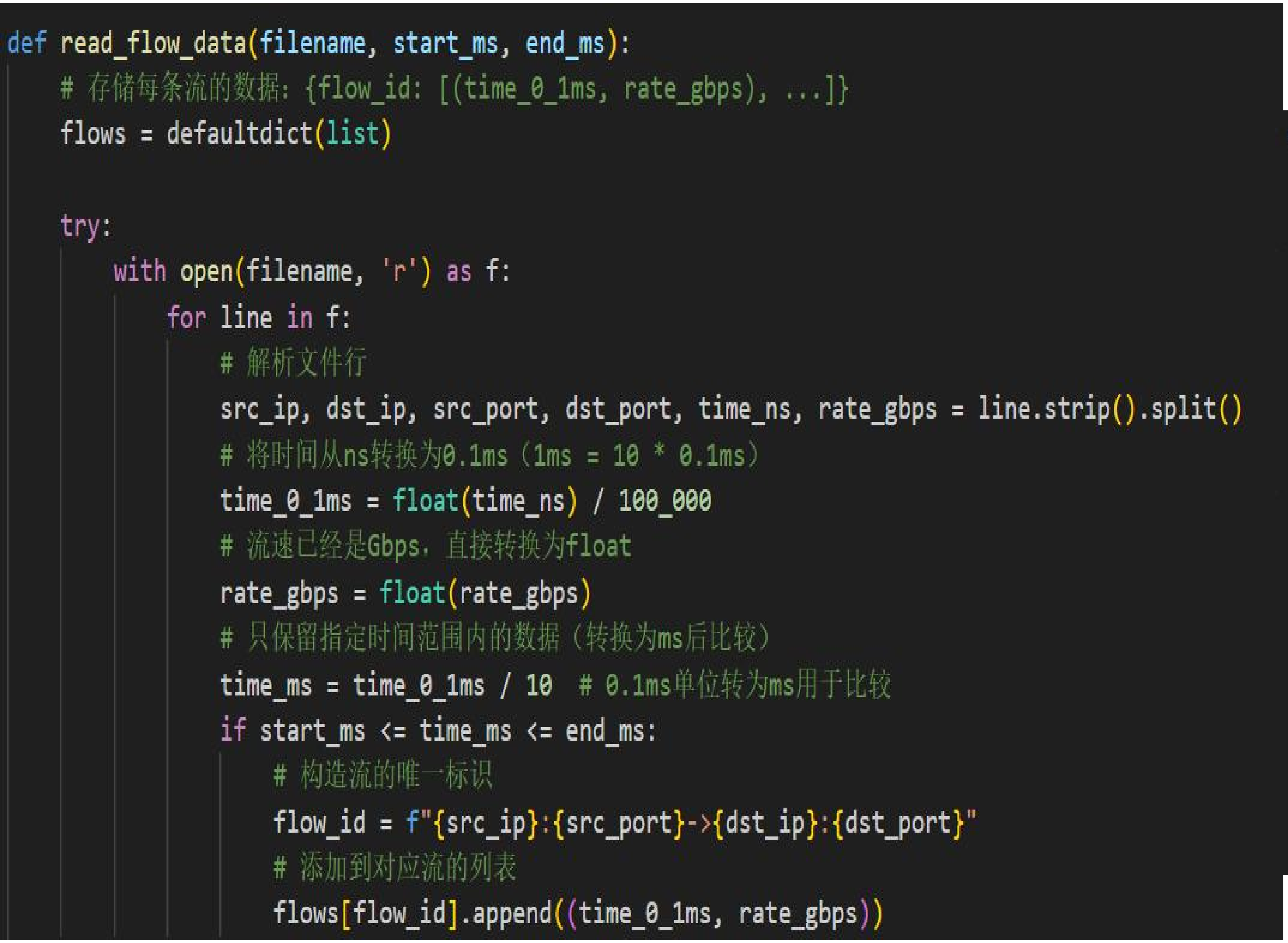
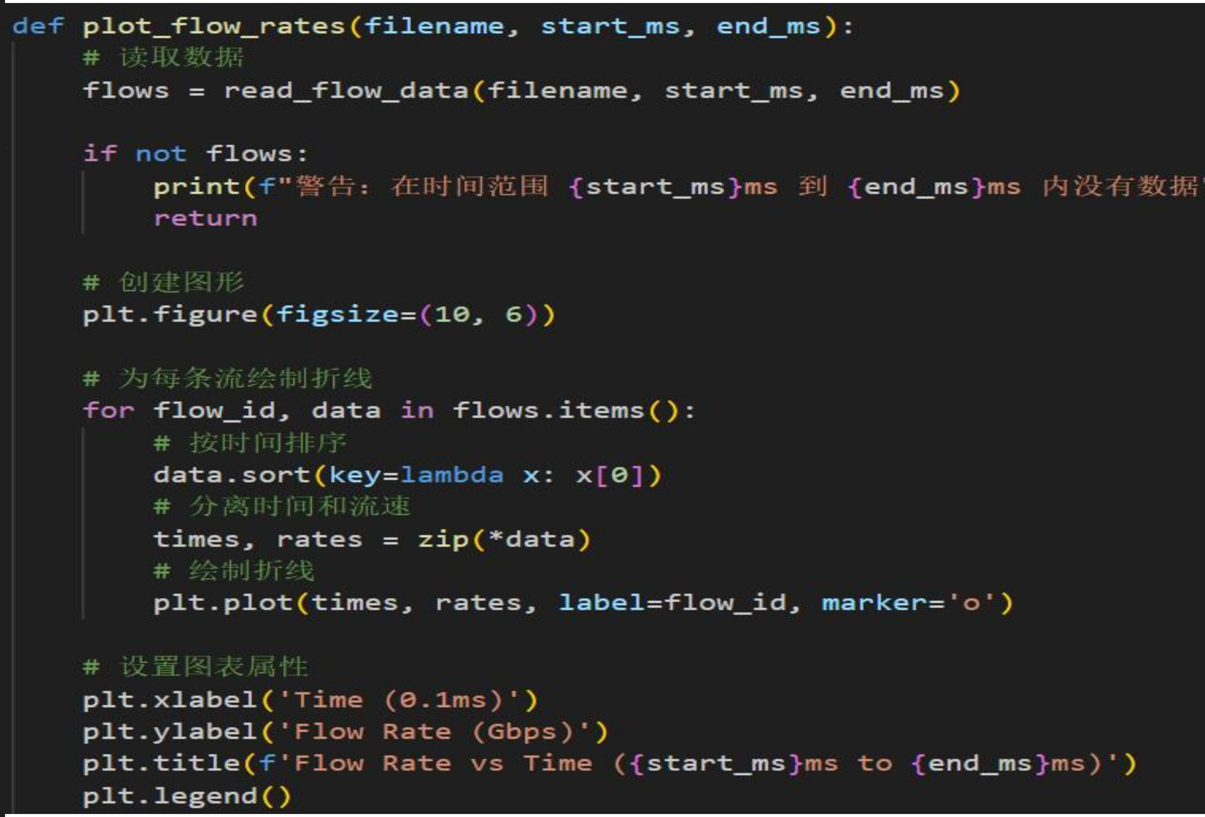
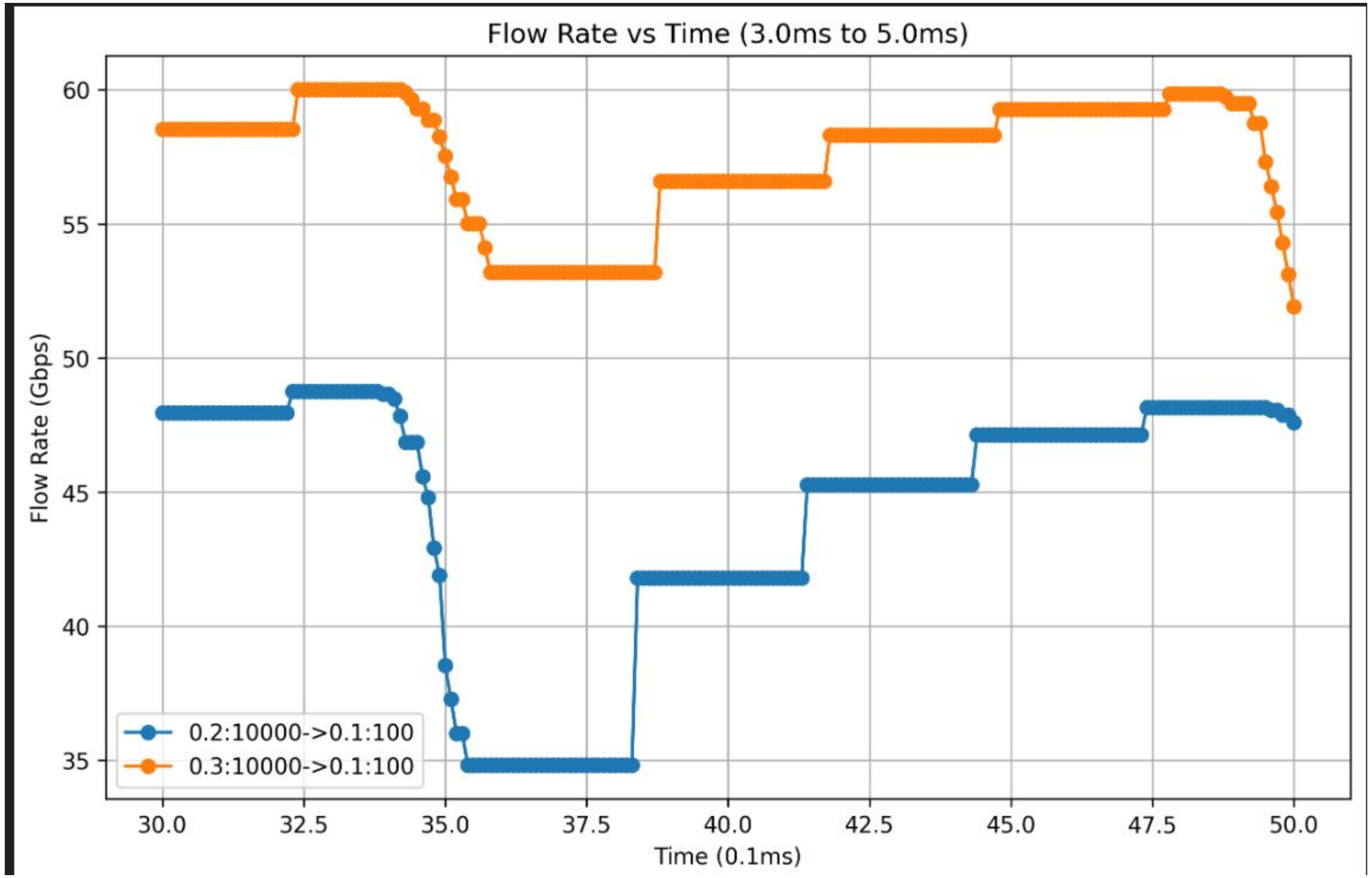


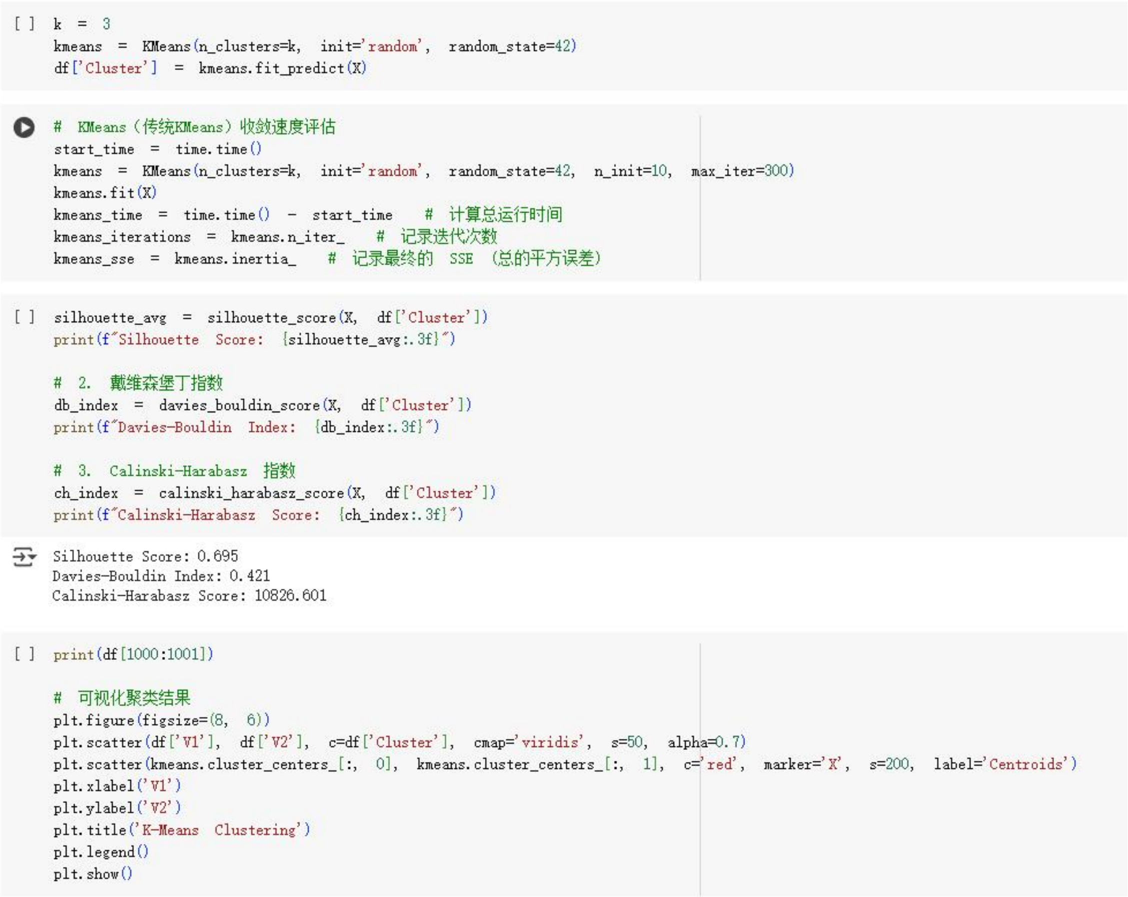
```
def read_flow_data(filename, start_ms, end_ms):
    # 存储每条流的数据：{flow_id: [(time_0_1ms, rate_gbps), ...]}
    flows = defaultdict(list)

    try:
        with open(filename, 'r') as f:
            for line in f:
                # 解析文件行
                src_ip, dst_ip, src_port, dst_port, time_ns, rate_gbps = line.strip().split()
                # 将时间从ns转换为0.1ms（1ms = 10 * 0.1ms）
                time_0_1ms = float(time_ns) / 100_000
                # 流速已经是Gbps，直接转换为float
                rate_gbps = float(rate_gbps)
                # 只保留指定时间范围内的数据（转换为ms后比较）
                time_ms = time_0_1ms / 10  # 0.1ms单位转为ms用于比较
                if start_ms <= time_ms <= end_ms:
                    # 构造流的唯一标识
                    flow_id = f"{src_ip}:{src_port}->{dst_ip}:{dst_port}"
                    # 添加到对应流的列表
                    flows[flow_id].append((time_0_1ms, rate_gbps))
```



```
def plot_flow_rates(filename, start_ms, end_ms):
    # 读取数据
    flows = read_flow_data(filename, start_ms, end_ms)

    if not flows:
        print(f"警告：在时间范围 {start_ms}ms 到 {end_ms}ms 内没有数据
        return

    # 创建图形
    plt.figure(figsize=(10, 6))

    # 为每条流绘制折线
    for flow_id, data in flows.items():
        # 按时间排序
        data.sort(key=lambda x: x[0])
        # 分离时间和流速
        times, rates = zip(*data)
        # 绘制折线
        plt.plot(times, rates, label=flow_id, marker='o')

    # 设置图表属性
    plt.xlabel('Time (0.1ms)')
    plt.ylabel('Flow Rate (Gbps)')
    plt.title(f'Flow Rate vs Time ({start_ms}ms to {end_ms}ms)')
    plt.legend()
```


Flow Rate vs Time (3.0ms to 5.0ms)
Flow Rate (Gbps)
60
55
50
45
40
35
30.0
32.5
35.0
37.5
40.0
42.5
45.0
47.5
50.0
Time (0.1ms)
0.2:10000->0.1:100
0.3:10000->0.1:100

流速变化图



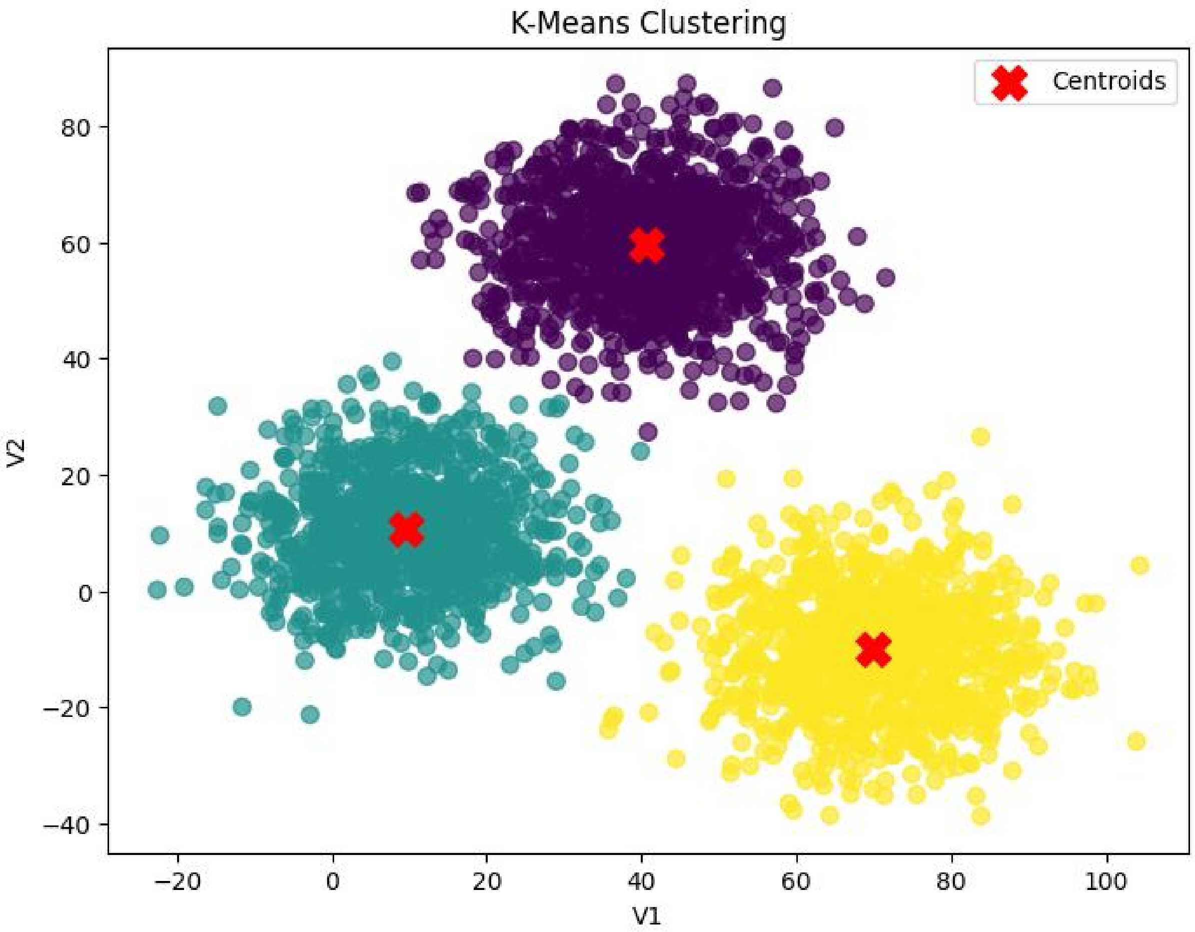
```
[ ] k = 3
    kmeans = KMeans(n_clusters=k, init='random', random_state=42)
    df['Cluster'] = kmeans.fit_predict(X)
```

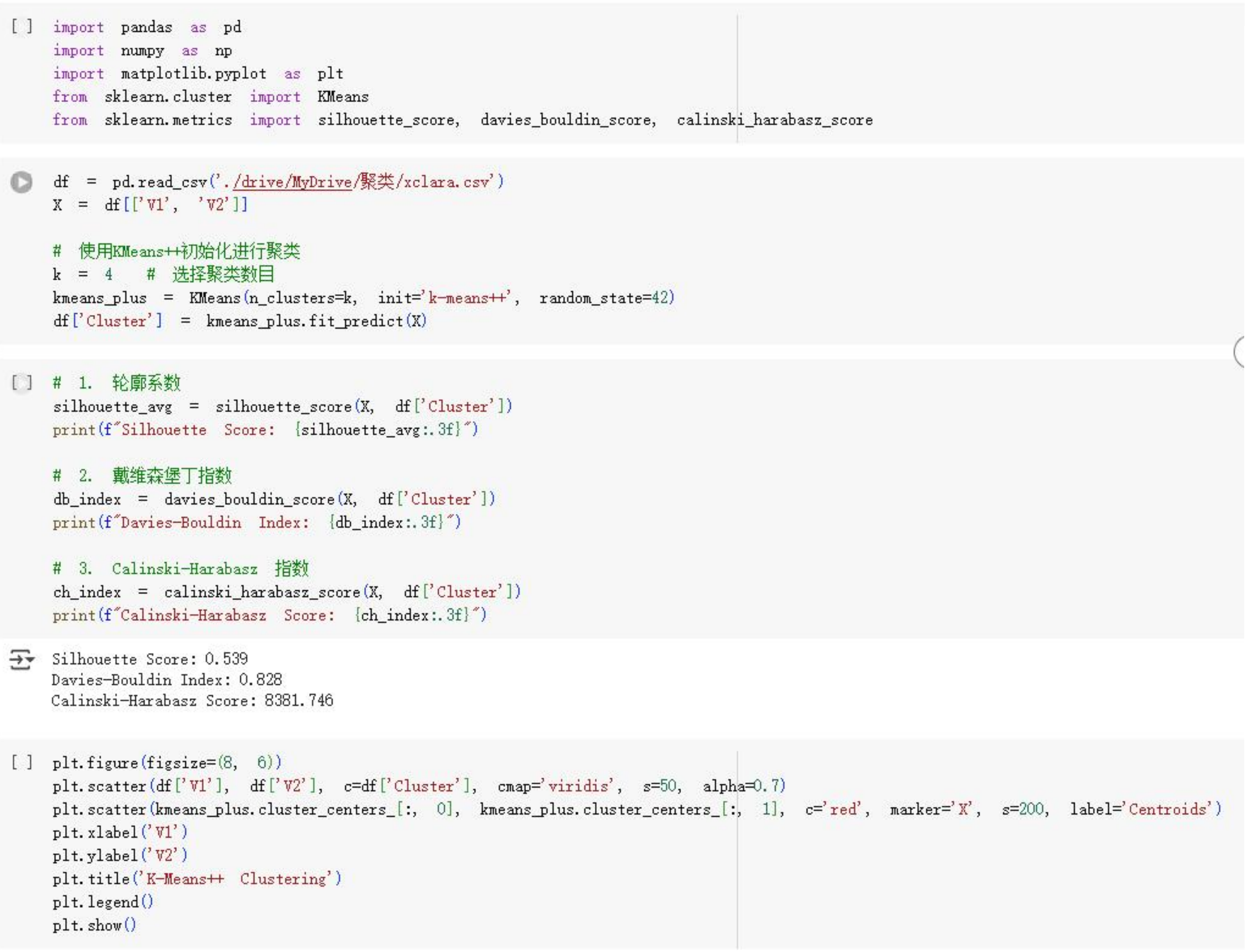
```
# KMeans（传统KMeans）收敛速度评估
start_time = time.time()
kmeans = KMeans(n_clusters=k, init='random', random_state=42, n_init=10, max_iter=300)
kmeans.fit(X)
kmeans_time = time.time() - start_time  # 计算总运行时间
kmeans_iterations = kmeans.n_iter_  # 记录迭代次数
kmeans_sse = kmeans.inertia_  # 记录最终的 SSE (总的平方误差)
```

```
[ ] silhouette_avg = silhouette_score(X, df['Cluster'])
    print(f"Silhouette Score: {silhouette_avg:.3f}")

    # 2. 戴维森堡丁指数
    db_index = davies_bouldin_score(X, df['Cluster'])
    print(f"Davies-Bouldin Index: {db_index:.3f}")

    # 3. Calinski-Harabasz 指数
    ch_index = calinski_harabasz_score(X, df['Cluster'])
    print(f"Calinski-Harabasz Score: {ch_index:.3f}")
```

```
Silhouette Score: 0.695
Davies-Bouldin Index: 0.421
Calinski-Harabasz Score: 10826.601
```

```
[ ] print(df[1000:1001])

    # 可视化聚类结果
    plt.figure(figsize=(8, 6))
    plt.scatter(df['V1'], df['V2'], c=df['Cluster'], cmap='viridis', s=50, alpha=0.7)
    plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='X', s=200, label='Centroids')
    plt.xlabel('V1')
    plt.ylabel('V2')
    plt.title('K-Means Clustering')
    plt.legend()
    plt.show()
```





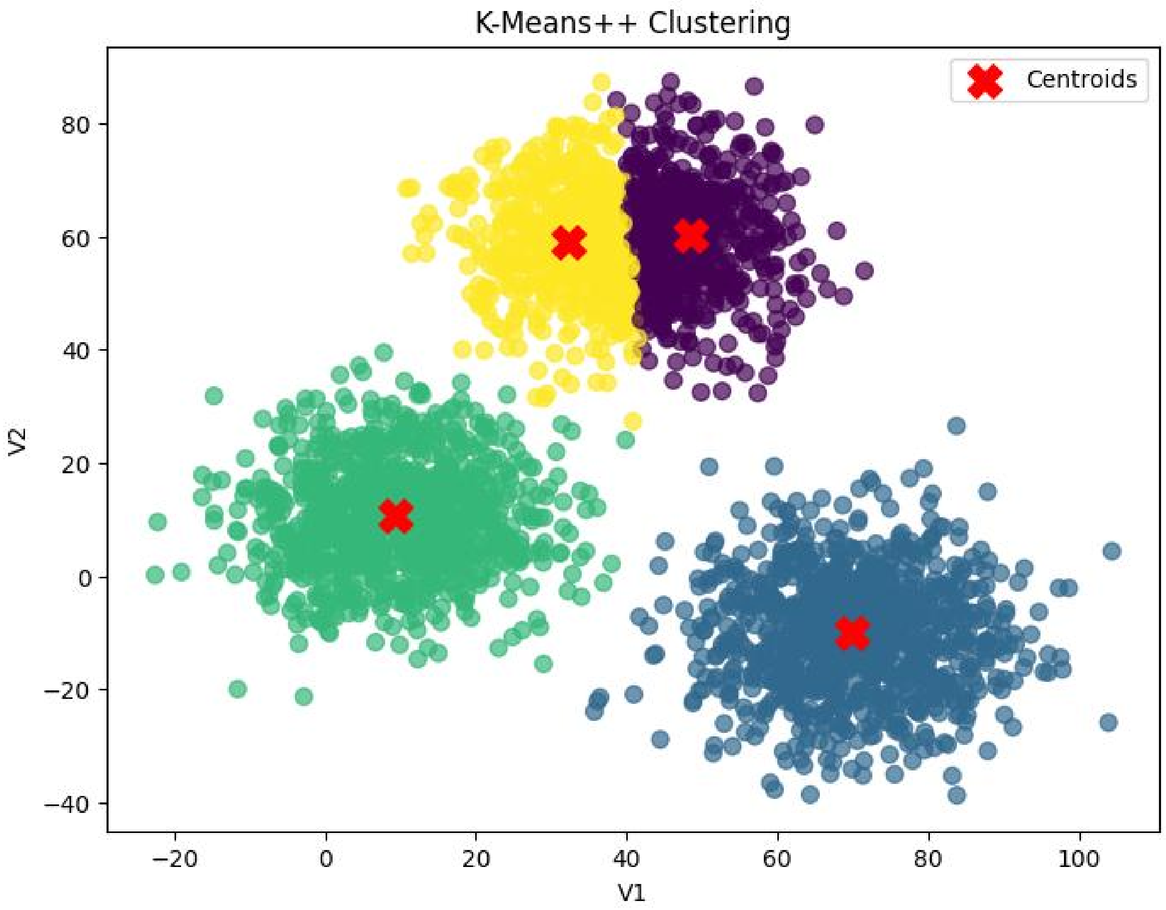
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, davies_bouldin_score, calinski_harabasz_score
```

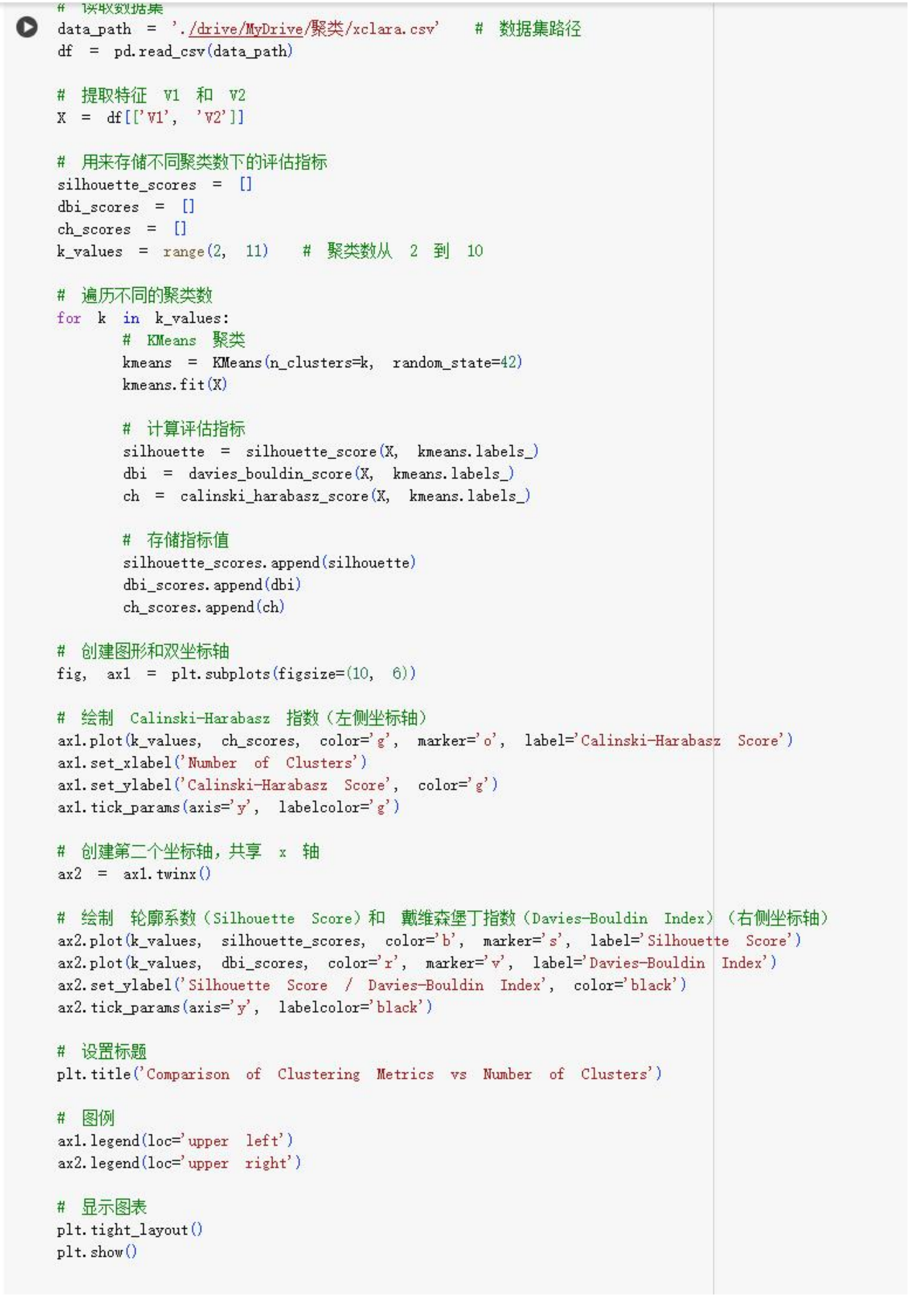
```
df = pd.read_csv('./drive/MyDrive/聚类/xclara.csv')
X = df[['V1', 'V2']]

# 使用KMeans++初始化进行聚类
k = 4  # 选择聚类数目
kmeans_plus = KMeans(n_clusters=k, init='k-means++', random_state=42)
df['Cluster'] = kmeans_plus.fit_predict(X)
```

```
# 1. 轮廓系数
silhouette_avg = silhouette_score(X, df['Cluster'])
print(f"Silhouette Score: {silhouette_avg:.3f}")

# 2. 戴维森堡丁指数
db_index = davies_bouldin_score(X, df['Cluster'])
print(f"Davies-Bouldin Index: {db_index:.3f}")

# 3. Calinski-Harabasz 指数
ch_index = calinski_harabasz_score(X, df['Cluster'])
print(f"Calinski-Harabasz Score: {ch_index:.3f}")
```

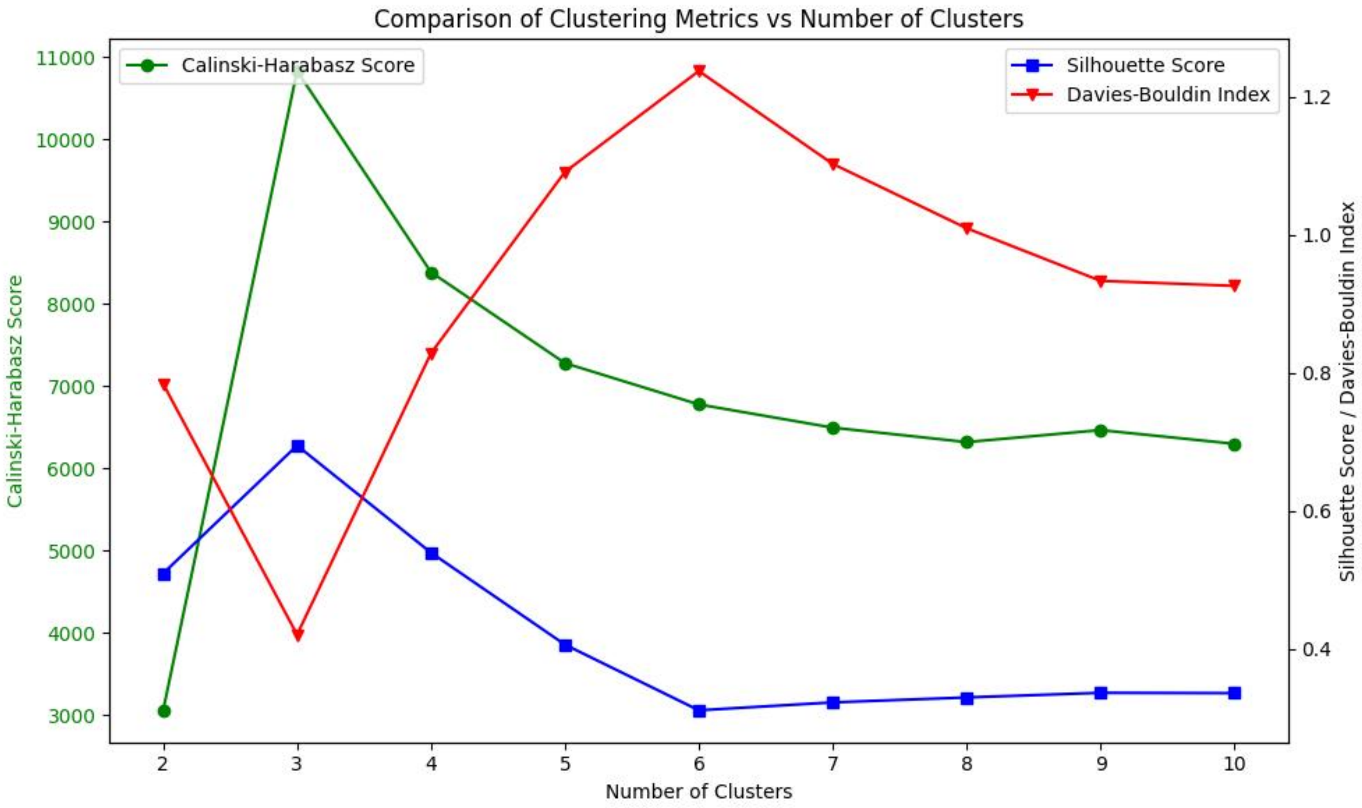
```
Silhouette Score: 0.539
Davies-Bouldin Index: 0.828
Calinski-Harabasz Score: 8381.746
```

```
plt.figure(figsize=(8, 6))
plt.scatter(df['V1'], df['V2'], c=df['Cluster'], cmap='viridis', s=50, alpha=0.7)
plt.scatter(kmeans_plus.cluster_centers_[:, 0], kmeans_plus.cluster_centers_[:, 1], c='red', marker='X', s=200, label='Centroids')
plt.xlabel('V1')
plt.ylabel('V2')
plt.title('K-Means++ Clustering')
plt.legend()
plt.show()
```





```python
# 读取数据集
data_path = './drive/MyDrive/聚类/xclara.csv'  # 数据集路径
df = pd.read_csv(data_path)

# 提取特征 V1 和 V2
X = df[['V1', 'V2']]

# 用来存储不同聚类数下的评估指标
silhouette_scores = []
dbi_scores = []
ch_scores = []
k_values = range(2, 11)  # 聚类数从 2 到 10

# 遍历不同的聚类数
for k in k_values:
        # KMeans 聚类
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X)

        # 计算评估指标
        silhouette = silhouette_score(X, kmeans.labels_)
        dbi = davies_bouldin_score(X, kmeans.labels_)
        ch = calinski_harabasz_score(X, kmeans.labels_)

        # 存储指标值
        silhouette_scores.append(silhouette)
        dbi_scores.append(dbi)
        ch_scores.append(ch)

# 创建图形和双坐标轴
fig, ax1 = plt.subplots(figsize=(10, 6))

# 绘制 Calinski-Harabasz 指数（左侧坐标轴）
ax1.plot(k_values, ch_scores, color='g', marker='o', label='Calinski-Harabasz Score')
ax1.set_xlabel('Number of Clusters')
ax1.set_ylabel('Calinski-Harabasz Score', color='g')
ax1.tick_params(axis='y', labelcolor='g')

# 创建第二个坐标轴，共享 x 轴
ax2 = ax1.twinx()

# 绘制 轮廓系数（Silhouette Score）和 戴维森堡丁指数（Davies-Bouldin Index）（右侧坐标轴）
ax2.plot(k_values, silhouette_scores, color='b', marker='s', label='Silhouette Score')
ax2.plot(k_values, dbi_scores, color='r', marker='v', label='Davies-Bouldin Index')
ax2.set_ylabel('Silhouette Score / Davies-Bouldin Index', color='black')
ax2.tick_params(axis='y', labelcolor='black')

# 设置标题
plt.title('Comparison of Clustering Metrics vs Number of Clusters')

# 图例
ax1.legend(loc='upper left')
ax2.legend(loc='upper right')

# 显示图表
plt.tight_layout()
plt.show()
```


Comparison of Clustering Metrics vs Number of Clusters
Calinski-Harabasz Score
Silhouette Score
Davies-Bouldin Index
Calinski-Harabasz Score
Number of Clusters
Silhouette Score / Davies-Bouldin Index