C

2025 年 1 月 26 日

1 数据预处理

1.1 导入数据

[1]:
```
# 导入相关 package
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt
import chardet
```

[2]:
```
# 定义一个函数，自动检测文件编码并读取文件
def read_csv_with_detected_encoding(file_path):
    # 检测文件编码
    with open(file_path, 'rb') as f:
        result = chardet.detect(f.read())
        encoding = result['encoding']
        print(f"检测到文件 {file_path} 的编码格式为: {encoding}")
    # 使用检测到的编码读取文件
    return pd.read_csv(file_path, encoding=encoding)

# 读取 data_dictionary.csv 文件
csv_content =␣
 ↪read_csv_with_detected_encoding('2025_Problem_C_Data\\data_dictionary.csv')
print("data_dictionary.csv 数据预览: ")
print(csv_content.head())

# 读取 summerOly_medal_counts.csv 文件
```

```
medal_counts = read_csv_with_detected_encoding('2025_Problem_C_Data\\summerOly_medal_counts.csv')
print("\nsummerOly_medal_counts.csv 数据预览: ")
print(medal_counts.head())

# 读取 summerOly_hosts.csv 文件
olympic_hosts = read_csv_with_detected_encoding('2025_Problem_C_Data\\summerOly_hosts.csv')
print("\nsummerOly_hosts.csv 数据预览: ")
print(olympic_hosts.head())

# 读取 summerOly_programs.csv 文件
olympic_programs = read_csv_with_detected_encoding('2025_Problem_C_Data\\summerOly_programs.csv')
print("\nsummerOly_programs.csv 数据预览: ")
print(olympic_programs.head())

# 读取 summerOly_athletes.csv 文件
olympic_athletes = read_csv_with_detected_encoding('2025_Problem_C_Data\\summerOly_athletes.csv')
print("\nsummerOly_athletes.csv 数据预览: ")
print(olympic_athletes.head())
```

```
检测到文件 2025_Problem_C_Data\data_dictionary.csv 的编码格式为: Windows-1252
data_dictionary.csv 数据预览:
  summerOly_medal_counts.csv                                  Unnamed: 1  \
0                  variables                                 explanation
1                       Rank      Rank of country based on total medals won
2                        NOC  Name of country as recorded for that Olympics
3                       Gold       Number of Gold medals the country earned
4                     Silver     Number of Silver medals the country earned

      Unnamed: 2
```

```
0        example
1           1, 2
2  China, France
3        0, 1, 2
4        0, 1, 2
```

检测到文件 2025_Problem_C_Data\summerOly_medal_counts.csv 的编码格式为: utf-8

summerOly_medal_counts.csv 数据预览:

	Rank	NOC	Gold	Silver	Bronze	Total	Year
0	1	United States	11	7	2	20	1896
1	2	Greece	10	18	19	47	1896
2	3	Germany	6	5	2	13	1896
3	4	France	5	4	2	11	1896
4	5	Great Britain	2	3	2	7	1896

检测到文件 2025_Problem_C_Data\summerOly_hosts.csv 的编码格式为: UTF-8-SIG

summerOly_hosts.csv 数据预览:

	Year	Host
0	1896	Athens, Greece
1	1900	Paris, France
2	1904	St. Louis, United States
3	1908	London, United Kingdom
4	1912	Stockholm, Sweden

检测到文件 2025_Problem_C_Data\summerOly_programs.csv 的编码格式为: Windows-1252

summerOly_programs.csv 数据预览:

	Sport	Discipline	Code	Sports Governing Body	1896	1900	1904	\
0	Aquatics	Artistic Swimming	SWA	World Aquatics	0	0	0	
1	Aquatics	Diving	DIV	World Aquatics	0	0	2	
2	Aquatics	Marathon Swimming	OWS	World Aquatics	0	0	0	
3	Aquatics	Swimming	SWM	World Aquatics	4	7	9	
4	Aquatics	Water Polo	WPO	World Aquatics	0	1	1	

	1906*	1908	1912	…	1988	1992	1996	2000	2004	2008	2012	2016	2020	\
0	0	0	0	…	2	2	1.0	2.0	2.0	2.0	2.0	2.0	2.0	
1	1	2	4	…	4	4	4.0	8.0	8.0	8.0	8.0	8.0	8.0	
2	0	0	0	…	0	0	0.0	0.0	0.0	2.0	2.0	2.0	2.0	

```
3      4     6    9  …    31   31  32.0  32.0  32.0  32.0  32.0  32.0  35.0
4      0     1    1  …     1    1   1.0   2.0   2.0   2.0   2.0   2.0   2.0

   2024
0   2.0
1   8.0
2   2.0
3  35.0
4   2.0

[5 rows x 35 columns]
检测到文件 2025_Problem_C_Data\summerOly_athletes.csv 的编码格式为: utf-8

summerOly_athletes.csv 数据预览:
                    Name Sex            Team  NOC  Year         City  \
0             A Dijiang   M           China  CHN  1992    Barcelona
1              A Lamusi   M           China  CHN  2012       London
2           Gunnar Aaby   M         Denmark  DEN  1920    Antwerpen
3           Edgar Aabye   M  Denmark/Sweden  DEN  1900        Paris
4  Cornelia (-strannood)   F     Netherlands  NED  1932  Los Angeles

        Sport                          Event     Medal
0  Basketball   Basketball Men's Basketball  No medal
1        Judo  Judo Men's Extra-Lightweight  No medal
2    Football       Football Men's Football  No medal
3  Tug-Of-War   Tug-Of-War Men's Tug-Of-War      Gold
4   Athletics  Athletics Women's 100 metres  No medal
```

1.2 数据清洗

1.2.1 缺失值检查

[3]:
```python
# 1. 缺失值检查
def check_missing_values(file_path):
    """
    检查 CSV 文件中的缺失值。
```

```
    参数:
        file_path (str): CSV 文件的路径。

    返回:
        None, 但会打印缺失值的相关信息。
    """
    try:
        # 尝试读取 CSV 文件
        data = pd.read_csv(file_path, encoding='utf-8')
    except UnicodeDecodeError:
        data = pd.read_csv(file_path, encoding='ISO-8859-1')

    print(file_path)

    # 检查每列的缺失值数量
    missing_values_per_column = data.isnull().sum()
    print("每列的缺失值数量: ")
    print(missing_values_per_column)

    # 检查整个数据框的总缺失值数量
    total_missing_values = missing_values_per_column.sum()
    print("整个数据框的总缺失值数量: ", total_missing_values)

    # 检查是否有任何缺失值
    has_missing_values = data.isnull().values.any()
    print("数据框中是否存在缺失值: ", has_missing_values)
    print("\n")

    # 如果有缺失值, 输出包含缺失值的行
    if has_missing_values:
        print("\n包含缺失值的行: ")
        print(data[data.isnull().any(axis=1)])
```

```
content_name = ['2025_Problem_C_Data\\summerOly_medal_counts.csv',␣
 ↪'2025_Problem_C_Data\\summerOly_hosts.csv',␣
 ↪'2025_Problem_C_Data\\summerOly_programs.csv',␣
 ↪'2025_Problem_C_Data\\summerOly_athletes.csv']
for i in content_name:
    check_missing_values(i)
```

```
2025_Problem_C_Data\summerOly_medal_counts.csv
每列的缺失值数量:
Rank      0
NOC       0
Gold      0
Silver    0
Bronze    0
Total     0
Year      0
dtype: int64
整个数据框的总缺失值数量:  0
数据框中是否存在缺失值:  False


2025_Problem_C_Data\summerOly_hosts.csv
每列的缺失值数量:
Year    0
Host    0
dtype: int64
整个数据框的总缺失值数量:  0
数据框中是否存在缺失值:  False


2025_Problem_C_Data\summerOly_programs.csv
每列的缺失值数量:
Sport                    0
Discipline               2
Code                     0
Sports Governing Body    0
1896                     0
```

```
1900                    0
1904                    0
1906*                   0
1908                    0
1912                    0
1920                    0
1924                    0
1928                    2
1932                    2
1936                    2
1948                    2
1952                    2
1956                    2
1960                    2
1964                    2
1968                    2
1972                    2
1976                    2
1980                    2
1984                    2
1988                    3
1992                    2
1996                    2
2000                    2
2004                    2
2008                    2
2012                    2
2016                    2
2020                    2
2024                    2
dtype: int64
整个数据框的总缺失值数量： 49
数据框中是否存在缺失值： True
```

包含缺失值的行：

```
                Sport     Discipline Code Sports Governing Body 1896 1900  \
12      Basque Pelota  Basque Pelota  PEL                  FIPV    0    1
44  Modern Pentathlon            NaN  MPN                  UIPM    0    0
65  Water Motorsports            NaN  PBT                   UIM    0
69            Skating         Figure  FSK                   ISU    0    0
70         Ice Hockey     Ice Hockey  IHO                  IIHF    0    0

   1904  1906*  1908 1912  …  1988 1992 1996 2000 2004 2008  2012 2016  \
12    0      0     0    0  …   NaN       0.0  0.0  0.0  0.0   0.0  0.0
44    0      0     0    1  …     2    2  1.0  2.0  2.0  2.0   2.0  2.0
65    0      0     3    0  …     0    0  0.0  0.0  0.0  0.0   0.0  0.0
69    0      0     4    0  …   NaN  NaN  NaN  NaN  NaN  NaN   NaN  NaN
70    0      0     0    0  …   NaN  NaN  NaN  NaN  NaN  NaN   NaN  NaN

    2020 2024
12   0.0  0.0
44   2.0  2.0
65   0.0  0.0
69   NaN  NaN
70   NaN  NaN

[5 rows x 35 columns]
2025_Problem_C_Data\summerOly_athletes.csv
每列的缺失值数量:
Name     0
Sex      0
Team     0
NOC      0
Year     0
City     0
Sport    0
Event    0
Medal    0
dtype: int64
整个数据框的总缺失值数量： 0
数据框中是否存在缺失值： False
```

1.2.2 补全 summerOly_programs.csv 中的缺失值

[4]:
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsRegressor
import re
import os

# 确保保存结果的目录存在
os.makedirs('Generated', exist_ok=True)

data = olympic_programs.copy()

# 3. 检查缺失值
#print(" 每列的缺失值数量: ")
#print(data.isnull().sum())

# 4. 填充 Discipline 列的缺失值
data['Discipline'] = data['Discipline'].fillna(data['Sport'])

# 5. 准备年份列的数据
years = [col for col in data.columns if col.isdigit() or col.endswith('*')]

# 6. 将数据从宽格式转换为长格式
data_long = data.melt(id_vars=['Sport', 'Discipline', 'Code', 'Sports Governing␣
 ↪Body'],
                      value_vars=years,
                      var_name='Year',
                      value_name='Events')

# 7. 将年份列转换为数值
data_long['Year'] = data_long['Year'].str.replace('*', '').astype(int)
```

```python
# 8. 清理 Events 列中的非数值字符
def clean_events(value):
    if isinstance(value, str):
        # 移除非数值字符
        cleaned_value = re.sub(r'[^0-9]', '', value)
        return float(cleaned_value) if cleaned_value.isdigit() else np.nan
    return value

data_long['Events'] = data_long['Events'].apply(clean_events)

# 9. 将 1924 年以及之后的 Skating 和 Ice Hockey 项目的赛事数目填为 0
mask = (data_long['Year'] >= 1924) & (data_long['Sport'].isin(['Skating', 'Ice␣
↪Hockey']))
data_long.loc[mask, 'Events'] = 0

# 10. 分组处理，按运动种类单独训练模型
for sport, group in data_long.groupby('Sport'):
    # 分离已知数据和缺失数据
    known_data = group.dropna(subset=['Events'])
    missing_data = group[group['Events'].isna()]

    if not known_data.empty and not missing_data.empty:
        # 准备训练数据
        X_known = known_data[['Year']]
        y_known = known_data['Events']

        # 检查已知数据的数量
        if len(y_known) < 5:
            print(f"警告: 运动种类 '{sport}' 的已知数据太少，使用 KNN 或线性回归填
充。")

            # 尝试使用线性回归
            if len(y_known) >= 3:  # 至少需要 3 个点来拟合线性回归
                model = LinearRegression()
                model.fit(X_known, y_known)
                predicted_events = model.predict(missing_data[['Year']])
```

```
            else:  # 使用 KNN, K=1
                model = KNeighborsRegressor(n_neighbors=1)
                model.fit(X_known, y_known)
                predicted_events = model.predict(missing_data[['Year']])

            # 将预测值四舍五入为整数
            predicted_events = np.round(predicted_events).astype(int)

            # 将预测值转换为 Pandas Series, 并确保索引对齐
            predicted_series = pd.Series(predicted_events, index=missing_data.
↪index)

            # 填充缺失值
            data_long.loc[data_long['Sport'] == sport, 'Events'] = data_long.
↪loc[data_long['Sport'] == sport, 'Events'].fillna(predicted_series)
        else:
            # 训练随机森林模型
            model = RandomForestRegressor(n_estimators=100, random_state=42)
            model.fit(X_known, y_known)

            # 预测缺失数据
            X_missing = missing_data[['Year']]
            predicted_events = model.predict(X_missing)

            # 将预测值四舍五入为整数
            predicted_events = np.round(predicted_events).astype(int)

            # 将预测值转换为 Pandas Series, 并确保索引对齐
            predicted_series = pd.Series(predicted_events, index=missing_data.
↪index)

            # 填充缺失值
            data_long.loc[data_long['Sport'] == sport, 'Events'] = data_long.
↪loc[data_long['Sport'] == sport, 'Events'].fillna(predicted_series)

            # 记录日志
```

```
            print(f"运动种类 '{sport}' 的模型训练完成，预测了␣
 ↪{len(predicted_events)} 个缺失值。")
    else:
        print(f"运动种类 '{sport}' 没有缺失数据或没有足够的已知数据。")

# 11. 将数据重新转换为宽格式
data_filled = data_long.pivot_table(index=['Sport', 'Discipline', 'Code',␣
 ↪'Sports Governing Body'],
                                    columns='Year',
                                    values='Events',
                                    aggfunc='first').reset_index()

# 12. 输出结果
print("\n填充后的数据：")
print(data_filled.head())

# 13. 保存结果到新的 CSV 文件
output_path = 'Generated\\summerOly_programs_filled.csv'
data_filled.to_csv(output_path, index=False, encoding='utf-8')  # 确保保存时使用
正确的编码
print(f"填充后的数据已保存到 {output_path}")
```

```
运动种类 'Aquatics' 没有缺失数据或没有足够的已知数据。
运动种类 'Archery' 没有缺失数据或没有足够的已知数据。
运动种类 'Athletics' 没有缺失数据或没有足够的已知数据。
运动种类 'Badminton' 的模型训练完成，预测了 2 个缺失值。
运动种类 'Baseball and Softball' 的模型训练完成，预测了 8 个缺失值。
运动种类 'Basketball' 的模型训练完成，预测了 2 个缺失值。
运动种类 'Basque Pelota' 的模型训练完成，预测了 4 个缺失值。
运动种类 'Boxing' 没有缺失数据或没有足够的已知数据。
运动种类 'Breaking' 没有缺失数据或没有足够的已知数据。
运动种类 'Canoeing' 的模型训练完成，预测了 1 个缺失值。
运动种类 'Cricket' 没有缺失数据或没有足够的已知数据。
运动种类 'Croquet' 没有缺失数据或没有足够的已知数据。
运动种类 'Cycling' 没有缺失数据或没有足够的已知数据。
运动种类 'Equestrian' 没有缺失数据或没有足够的已知数据。
运动种类 'Fencing' 没有缺失数据或没有足够的已知数据。
```

运动种类 'Field hockey' 没有缺失数据或没有足够的已知数据。
运动种类 'Flag football' 没有缺失数据或没有足够的已知数据。
运动种类 'Football' 没有缺失数据或没有足够的已知数据。
运动种类 'Golf' 没有缺失数据或没有足够的已知数据。
运动种类 'Gymnastics' 没有缺失数据或没有足够的已知数据。
运动种类 'Handball' 的模型训练完成，预测了 1 个缺失值。
运动种类 'Ice Hockey' 没有缺失数据或没有足够的已知数据。
运动种类 'Jeu de Paume' 没有缺失数据或没有足够的已知数据。
运动种类 'Judo' 没有缺失数据或没有足够的已知数据。
运动种类 'Karate' 没有缺失数据或没有足够的已知数据。
运动种类 'Lacrosse' 的模型训练完成，预测了 3 个缺失值。
运动种类 'Modern Pentathlon' 没有缺失数据或没有足够的已知数据。
运动种类 'Polo' 没有缺失数据或没有足够的已知数据。
运动种类 'Rackets' 没有缺失数据或没有足够的已知数据。
运动种类 'Roque' 没有缺失数据或没有足够的已知数据。
运动种类 'Rowing' 没有缺失数据或没有足够的已知数据。
运动种类 'Rugby' 没有缺失数据或没有足够的已知数据。
运动种类 'Sailing' 没有缺失数据或没有足够的已知数据。
运动种类 'Shooting' 没有缺失数据或没有足够的已知数据。
运动种类 'Skateboarding' 没有缺失数据或没有足够的已知数据。
运动种类 'Skating' 没有缺失数据或没有足够的已知数据。
运动种类 'Sport Climbing' 没有缺失数据或没有足够的已知数据。
运动种类 'Squash' 没有缺失数据或没有足够的已知数据。
运动种类 'Surfing' 没有缺失数据或没有足够的已知数据。
运动种类 'Table Tennis' 没有缺失数据或没有足够的已知数据。
运动种类 'Taekwondo' 的模型训练完成，预测了 2 个缺失值。
运动种类 'Tennis' 的模型训练完成，预测了 2 个缺失值。
运动种类 'Total disciplines' 没有缺失数据或没有足够的已知数据。
运动种类 'Total events' 没有缺失数据或没有足够的已知数据。
运动种类 'Total sports' 没有缺失数据或没有足够的已知数据。
运动种类 'Triathlon' 没有缺失数据或没有足够的已知数据。
运动种类 'Tug of War' 没有缺失数据或没有足够的已知数据。
运动种类 'Volleyball' 的模型训练完成，预测了 1 个缺失值。
运动种类 'Water Motorsports' 的模型训练完成，预测了 1 个缺失值。
运动种类 'Weightlifting' 没有缺失数据或没有足够的已知数据。
运动种类 'Wrestling' 没有缺失数据或没有足够的已知数据。

填充后的数据:

```
Year     Sport         Discipline Code Sports Governing Body  1896  1900  \
0     Aquatics  Artistic Swimming  SWA        World Aquatics   0.0   0.0
1     Aquatics             Diving  DIV        World Aquatics   0.0   0.0
2     Aquatics  Marathon Swimming  OWS        World Aquatics   0.0   0.0
3     Aquatics           Swimming  SWM        World Aquatics   4.0   7.0
4     Aquatics         Water Polo  WPO        World Aquatics   0.0   1.0

Year  1904  1906  1908  1912  …  1988  1992  1996  2000  2004  2008  2012  \
0      0.0   0.0   0.0   0.0  …   2.0   2.0   1.0   2.0   2.0   2.0   2.0
1      2.0   1.0   2.0   4.0  …   4.0   4.0   4.0   8.0   8.0   8.0   8.0
2      0.0   0.0   0.0   0.0  …   0.0   0.0   0.0   0.0   0.0   2.0   2.0
3      9.0   4.0   6.0   9.0  …  31.0  31.0  32.0  32.0  32.0  32.0  32.0
4      1.0   0.0   1.0   1.0  …   1.0   1.0   1.0   2.0   2.0   2.0   2.0

Year  2016  2020  2024
0      2.0   2.0   2.0
1      8.0   8.0   8.0
2      2.0   2.0   2.0
3     32.0  35.0  35.0
4      2.0   2.0   2.0

[5 rows x 35 columns]
填充后的数据已保存到 Generated\summerOly_programs_filled.csv
```

1.2.3 Medal_counts 数据清洗

[5]:
```python
# 2. 数据清洗
# 确保数据的格式正确
data = medal_counts[['Year', 'NOC', 'Gold', 'Silver', 'Bronze', 'Total']]

# 3. 创建年份和国家的索引
years = data['Year'].unique()
noc = data['NOC'].unique()

# 4. 定义一个函数来生成表格
def generate_table(data, column_name):
```

```
    # 创建一个空的 DataFrame，以年份为列，国家为行
    table = pd.DataFrame(index=noc, columns=years)

    # 填充数据
    for index, row in data.iterrows():
        year = row['Year']
        country = row['NOC']
        value = row[column_name]
        table.at[country, year] = value

    # 推断数据类型并填充缺失值为 0
    table = table.infer_objects(copy=False).fillna(0).astype(int)

    return table

# 5. 生成金牌、银牌、铜牌和总数的表格
gold_table = generate_table(data, 'Gold')
silver_table = generate_table(data, 'Silver')
bronze_table = generate_table(data, 'Bronze')
total_table = generate_table(data, 'Total')

# 6. 保存到新的 CSV 文件
gold_table.to_csv('Generated\\summerOly_gold_summary.csv')
silver_table.to_csv('Generated\\summerOly_silver_summary.csv')
bronze_table.to_csv('Generated\\summerOly_bronze_summary.csv')
total_table.to_csv('Generated\\summerOly_total_summary.csv')
```

[6]:
```
# 7. 输出结果
print("金牌表格：")
print(gold_table)
```

金牌表格：

	1896	1900	1904	1908	1912	1920	1924	1928	1932	\
United States	11	19	76	23	26	41	45	22	0	
Greece	10	0	1	0	1	0	0	0	0	
Germany	6	4	4	3	5	0	0	10	0	
France	5	27	0	5	7	9	13	6	0	

Great Britain	2	15	1	56	10	14	9	3	0
…	…	…	…	…	…	…	…	…	…
Saint Lucia	0	0	0	0	0	0	0	0	0
Dominica	0	0	0	0	0	0	0	0	0
Albania	0	0	0	0	0	0	0	0	0
Cabo Verde	0	0	0	0	0	0	0	0	0
Refugee Olympic Team	0	0	0	0	0	0	0	0	0

	1936	…	1988	1992	1996	2000	2004	2008	2012	\
United States	24	…	36	37	44	37	36	36	48	
Greece	0	…	0	2	4	4	6	0	0	
Germany	38	…	0	33	20	13	13	16	11	
France	7	…	6	8	15	13	11	7	11	
Great Britain	4	…	5	5	1	11	9	19	29	
…	…	…	…	…	…	…	…	…	…	
Saint Lucia	0	…	0	0	0	0	0	0	0	
Dominica	0	…	0	0	0	0	0	0	0	
Albania	0	…	0	0	0	0	0	0	0	
Cabo Verde	0	…	0	0	0	0	0	0	0	
Refugee Olympic Team	0	…	0	0	0	0	0	0	0	

	2016	2020	2024
United States	46	39	40
Greece	3	2	1
Germany	17	10	12
France	10	10	16
Great Britain	27	22	14
…	…	…	…
Saint Lucia	0	0	1
Dominica	0	0	1
Albania	0	0	0
Cabo Verde	0	0	0
Refugee Olympic Team	0	0	0

[210 rows x 30 columns]

[7]:
```
print("\n银牌表格: ")
print(silver_table)
```

```
银牌表格:
                        1896  1900  1904  1908  1912  1920  1924  1928  1932  \
United States              7    14    78    12    19    27    27    18     0
Greece                    18     0     0     3     0     1     0     0     0
Germany                    5     3     5     5    13     0     0     7     0
France                     4    39     1     5     4    19    15    10     0
Great Britain              3     7     1    51    15    15    13    10     0
…                        …     …     …     …     …     …     …     …     …
Saint Lucia                0     0     0     0     0     0     0     0     0
Dominica                   0     0     0     0     0     0     0     0     0
Albania                    0     0     0     0     0     0     0     0     0
Cabo Verde                 0     0     0     0     0     0     0     0     0
Refugee Olympic Team       0     0     0     0     0     0     0     0     0

                        1936  …  1988  1992  1996  2000  2004  2008  2012  \
United States             21  …    31    34    32    24    39    39    26
Greece                     0  …     0     0     4     6     6     2     0
Germany                   31  …     0    21    18    17    16    11    20
France                     6  …     4     5     7    14     9    16    11
Great Britain              7  …    10     3     8    10     9    13    18
…                        …  …     …     …     …     …     …     …     …
Saint Lucia                0  …     0     0     0     0     0     0     0
Dominica                   0  …     0     0     0     0     0     0     0
Albania                    0  …     0     0     0     0     0     0     0
Cabo Verde                 0  …     0     0     0     0     0     0     0
Refugee Olympic Team       0  …     0     0     0     0     0     0     0

                        2016  2020  2024
United States             37    41    44
Greece                     1     1     1
Germany                   10    11    13
France                    18    12    26
Great Britain             23    20    22
```

```
…                       …   …   …
Saint Lucia             0   0   1
Dominica                0   0   0
Albania                 0   0   0
Cabo Verde              0   0   0
Refugee Olympic Team    0   0   0

[210 rows x 30 columns]
```

[8]:
```
print("\n铜牌表格: ")
print(bronze_table)
```

```
铜牌表格:
                        1896  1900  1904  1908  1912  1920  1924  1928  1932  \
United States              2    15    77    12    19    27    27    16     0
Greece                    19     0     1     1     1     0     0     0     0
Germany                    2     2     6     5     7     0     0    14     0
France                     2    37     0     9     3    13    10     5     0
Great Britain              2     9     0    39    16    13    12     7     0
…                        …     …     …     …     …     …     …     …     …
Saint Lucia                0     0     0     0     0     0     0     0     0
Dominica                   0     0     0     0     0     0     0     0     0
Albania                    0     0     0     0     0     0     0     0     0
Cabo Verde                 0     0     0     0     0     0     0     0     0
Refugee Olympic Team       0     0     0     0     0     0     0     0     0

                        1936  …  1988  1992  1996  2000  2004  2008  2012  \
United States             12  …    27    37    25    32    26    37    30
Greece                     0  …     1     0     0     3     4     1     2
Germany                   32  …     0    28    27    26    20    14    13
France                     6  …     6    16    15    11    13    20    13
Great Britain              3  …     9    12     6     7    12    19    18
…                        …  …     …     …     …     …     …     …     …
Saint Lucia                0  …     0     0     0     0     0     0     0
Dominica                   0  …     0     0     0     0     0     0     0
Albania                    0  …     0     0     0     0     0     0     0
```

```
Cabo Verde                0  …     0     0     0     0     0     0     0
Refugee Olympic Team      0  …     0     0     0     0     0     0     0

                       2016  2020  2024
United States            38    33    42
Greece                    2     1     6
Germany                  15    16     8
France                   14    11    22
Great Britain            17    22    29
…                     …    …     …
Saint Lucia               0     0     0
Dominica                  0     0     0
Albania                   0     0     2
Cabo Verde                0     0     1
Refugee Olympic Team      0     0     1

[210 rows x 30 columns]
```

[9]:
```
print("\n总数表格: ")
print(total_table)
```

```
总数表格:
                       1896  1900  1904  1908  1912  1920  1924  1928  1932  \
United States            20    48   231    47    64    95    99    56     0
Greece                   47     0     2     4     2     1     0     0     0
Germany                  13     9    15    13    25     0     0    31     0
France                   11   103     1    19    14    41    38    21     0
Great Britain             7    31     2   146    41    42    34    20     0
…                     …    …    …    …    …    …    …    …    …
Saint Lucia               0     0     0     0     0     0     0     0     0
Dominica                  0     0     0     0     0     0     0     0     0
Albania                   0     0     0     0     0     0     0     0     0
Cabo Verde                0     0     0     0     0     0     0     0     0
Refugee Olympic Team      0     0     0     0     0     0     0     0     0

                       1936  …  1988  1992  1996  2000  2004  2008  2012  \
```

```
United States            57  …    94   108   101    93   101   112   104
Greece                    0  …     1     2     8    13    16     3     2
Germany                 101  …     0    82    65    56    49    41    44
France                   19  …    16    29    37    38    33    43    35
Great Britain            14  …    24    20    15    28    30    51    65
…                     …  …   …   …   …   …   …   …   …
Saint Lucia               0  …     0     0     0     0     0     0     0
Dominica                  0  …     0     0     0     0     0     0     0
Albania                   0  …     0     0     0     0     0     0     0
Cabo Verde                0  …     0     0     0     0     0     0     0
Refugee Olympic Team      0  …     0     0     0     0     0     0     0

                       2016  2020  2024
United States           121   113   126
Greece                    6     4     8
Germany                  42    37    33
France                   42    33    64
Great Britain            67    64    65
…                     …  …   …
Saint Lucia               0     0     2
Dominica                  0     0     1
Albania                   0     0     2
Cabo Verde                0     0     1
Refugee Olympic Team      0     0     1

[210 rows x 30 columns]
```

1.2.4 清理 athletes.csv 并转换格式为宽

[10]:
```
# 读取 summerOly_athletes.csv 文件
data = olympic_athletes.copy()

# 转换为长格式，将年份放到列的抬头位置
pivot_df = data.pivot_table(index=['Name', 'Sex', 'Team', 'NOC', 'City',␣
 ↪'Sport', 'Event'],
                                columns='Year',
                                values='Medal',
```

```
                                aggfunc='first').reset_index()

# 填充缺失值为 0
pivot_df = pivot_df.fillna(0)

# 输出结果
print("转换为宽格式后的数据: ")
print(pivot_df.head())

# 保存为新的 CSV 文件
output_path = 'Generated\\summerOly_athletes_wide_format.csv'
pivot_df.to_csv(output_path, index=False, encoding='utf-8')
print(f"宽格式数据已保存到 {output_path}")
```

转换为宽格式后的数据:

```
Year           Name Sex          Team  NOC   City          Sport  \
0      (jr) Larocca   M     Argentina  ARG  Paris     Equestrian
1     . Chadalavada   F         India  IND  Tokyo        Fencing
2            . Deni   M     Indonesia  INA  Tokyo  Weightlifting
3               671   F         China  CHN  Paris       Breaking
4          A Alayed   F  Saudi Arabia  KSA  Paris       Swimming

Year                    Event 1896 1900 1904  … 1988 1992 1996 2000 2004  \
0           Jumping Individual    0    0    0  …    0    0    0    0    0
1     Women's Sabre Individual    0    0    0  …    0    0    0    0    0
2                   Men's 67kg    0    0    0  …    0    0    0    0    0
3                      B-Girls    0    0    0  …    0    0    0    0    0
4       Women's 200m Freestyle    0    0    0  …    0    0    0    0    0

Year 2008 2012 2016      2020      2024
0       0    0    0         0  No medal
1       0    0    0  No medal         0
2       0    0    0  No medal         0
3       0    0    0         0    Bronze
4       0    0    0         0  No medal

[5 rows x 38 columns]
```

宽格式数据已保存到 Generated\summerOly_athletes_wide_format.csv

2 分析数据

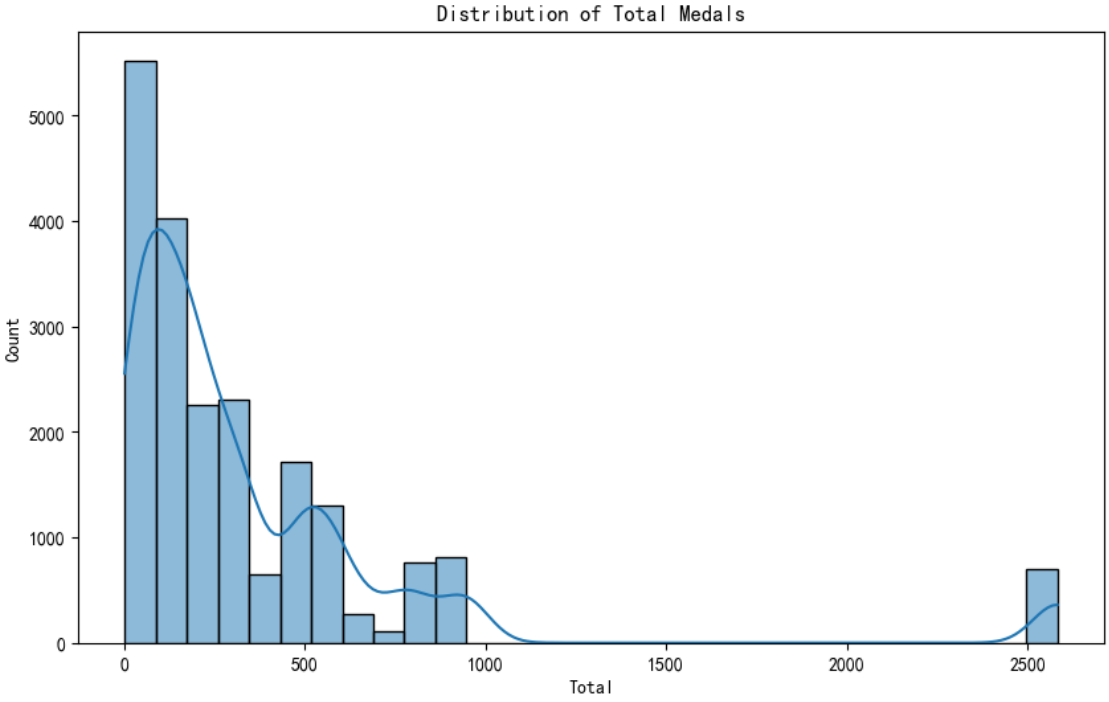
2.1 国家级特征

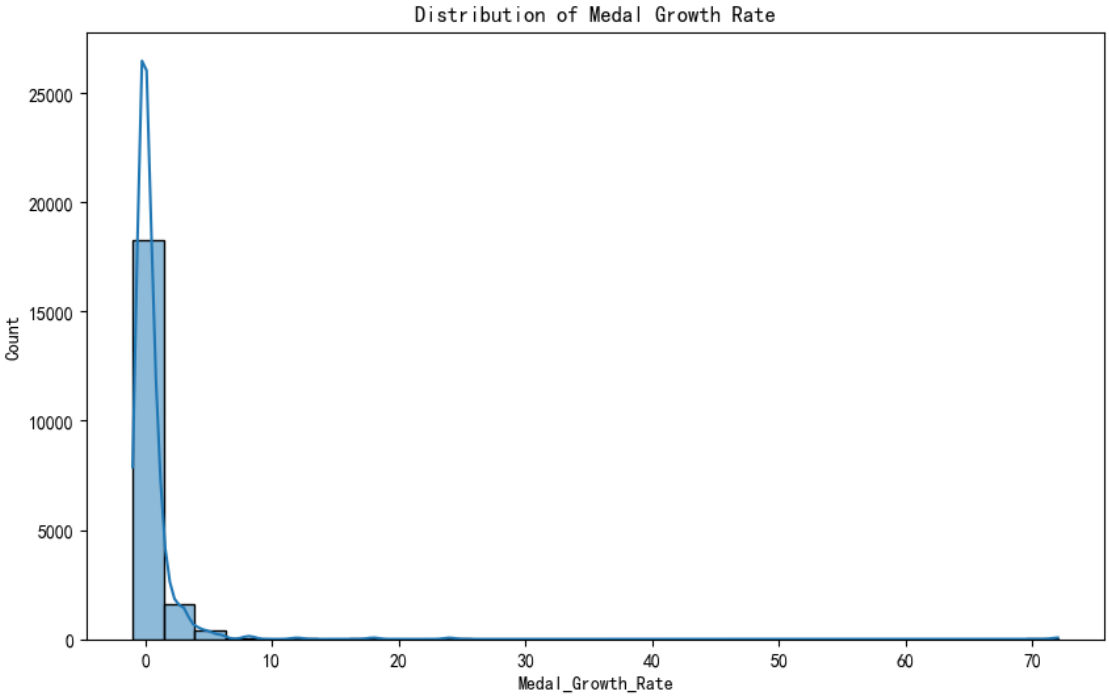
[33]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# 加载奖牌数据
medal_data = pd.read_csv('2025_Problem_C_Data\\summerOly_medal_counts.csv')

# 加载主办国数据
host_data = pd.read_csv('2025_Problem_C_Data\\summerOly_hosts.csv')

# 数据清洗
medal_data.fillna(0, inplace=True)

# 提取主办国信息
host_data['Host_NOC'] = host_data['Host'].str.split(', ').str[-1]  # 提取国家名
称
host_data = host_data[['Year', 'Host_NOC']].rename(columns={'Host_NOC': 'NOC'})

# 标记主办国
medal_data = pd.merge(medal_data, host_data, on=['Year', 'NOC'], how='left')
medal_data['Is_Host'] = medal_data['NOC'].notnull().astype(int)

# 计算奖牌增长率
def calculate_growth_rate(group):
    group = group.sort_values('Year')
    group['Medal_Growth_Rate'] = group['Total'].pct_change()
    return group

growth_data = medal_data.groupby('NOC').apply(calculate_growth_rate).
 ↪reset_index(drop=True)
growth_data = growth_data.dropna(subset=['Medal_Growth_Rate'])  # 删除 NaN 值
```

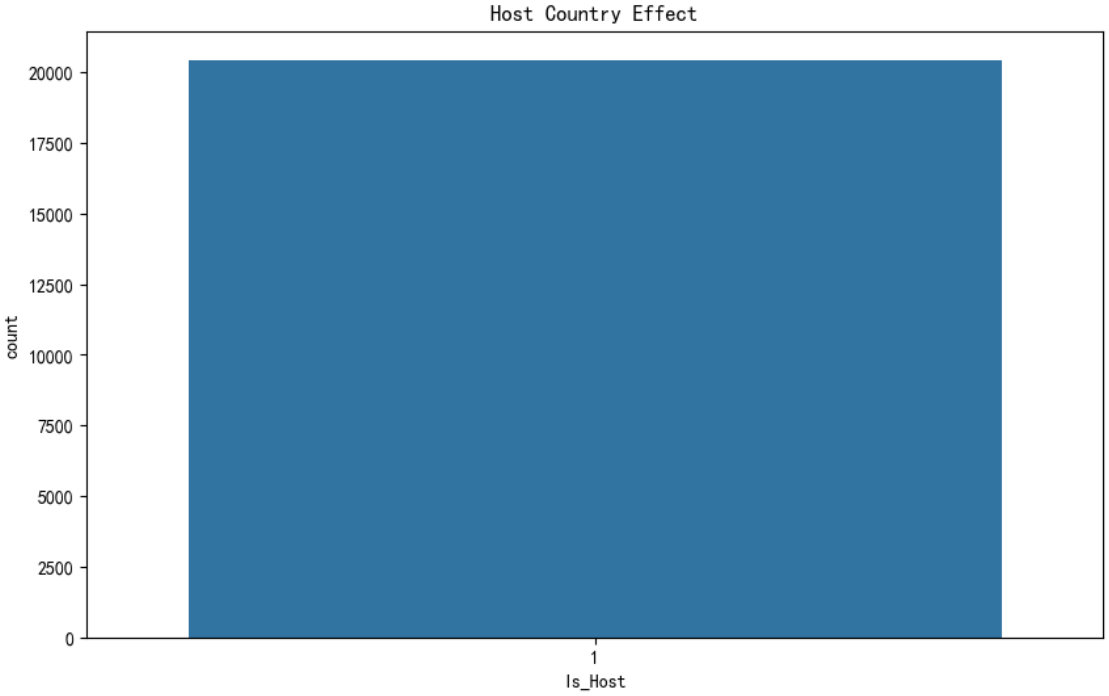
```
# 构建特征矩阵
medal_totals = medal_data.groupby('NOC')['Total'].sum().reset_index()
gold_totals = medal_data.groupby('NOC')['Gold'].sum().reset_index().
 ↪rename(columns={'Gold': 'Total_Gold'})

features = pd.merge(medal_totals, gold_totals, on='NOC')
features = pd.merge(features, growth_data[['NOC', 'Medal_Growth_Rate']],␣
 ↪on='NOC', how='left')
features = pd.merge(features, medal_data[['NOC', 'Is_Host']], on='NOC',␣
 ↪how='left')
features.fillna(0, inplace=True)

# 保存特征矩阵
features.to_csv('national_features.csv', index=False)

# 可视化特征
plt.figure(figsize=(10, 6))
sns.histplot(features['Total'], bins=30, kde=True)
plt.title('Distribution of Total Medals')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(features['Medal_Growth_Rate'], bins=30, kde=True)
plt.title('Distribution of Medal Growth Rate')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='Is_Host', data=features)
plt.title('Host Country Effect')
plt.show()
```

```
C:\Users\Ziqi\AppData\Local\Temp\ipykernel_13772\107591029.py:28:
DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns.
This behavior is deprecated, and in a future version of pandas the grouping
columns will be excluded from the operation. Either pass `include_groups=False`
to exclude the groupings or explicitly select the grouping columns after groupby
```

```
to silence this warning.
  growth_data =
medal_data.groupby('NOC').apply(calculate_growth_rate).reset_index(drop=True)
```




Distribution of Medal Growth Rate
Count
Medal_Growth_Rate
0
5000
10000
15000
20000
25000
0
10
20
30
40
50
60
70


Host Country Effect
count
Is_Host
0
2500
5000
7500
10000
12500
15000
17500
20000
1

2.2 项目级特征

2.3 运动员级特征

2.3.1 预处理

[11]:
```
# 读取 summerOly_athletes.csv 文件
data = olympic_athletes.copy()

# 提取必要的列
athlete_years = olympic_athletes[['Name', 'Sex', 'NOC', 'Team', 'Year',␣
 ↪'Sport', 'Event']].drop_duplicates()
# 合并 Name, Sex, NOC 列
athlete_years['Feature'] = athlete_years['Name'] + ', ' + athlete_years['Sex'] ␣
 ↪+ ', ' + athlete_years['NOC']

# 删除原始的 Name, Sex, NOC 列
#athlete_years = athlete_years.drop(columns=['Name', 'Sex', 'NOC'])

# 对每个运动员进行排序
athlete_years = athlete_years.sort_values(by=['Feature','Year'])
athlete_years.to_csv('Generated\\athlete_yesrs.csv', index=False,␣
 ↪encoding='utf-8')
```

2.3.2 添加唯一特征值

[12]:
```
import os

# 设置环境变量 LOKY_MAX_CPU_COUNT
os.environ["LOKY_MAX_CPU_COUNT"] = "8"  # 使用 CPU 核心数
```

[13]:
```
import pandas as pd
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

# 读取 CSV 文件
file_path = 'Generated\\athlete_yesrs.csv'  # 替换为你的文件路径
data = pd.read_csv(file_path)
```

```
# 显示原始数据的前几行
print("原始数据的前几行：")
print(data.head())

# 设置时间阈值
time_threshold_small = 12
time_threshold_large = 44

# 按 Feature 分组
grouped = data.groupby('Feature')

# 用于存储处理后的数据
processed_data = []

# 遍历每个分组
for feature, group in grouped:
    # 按 Year 排序
    group = group.sort_values(by='Year')

    # 初始化变量
    unique_feature_count = 0
    last_year = None

    # 遍历分组中的每条记录
    for index, row in group.iterrows():
        current_year = row['Year']

        # 判断是否为同一个运动员
        if last_year is not None:
            year_diff = current_year - last_year
            if year_diff > time_threshold_large:
                # 如果时间跨度大于 44 年，直接认为是不同运动员
                unique_feature_count += 1
            elif year_diff > time_threshold_small:
                # 如果时间跨度在 12 到 44 年之间，进行聚类分析
```

```
                features_cluster = group[['Year', 'Sport', 'Event']].
↪apply(lambda x: x.factorize()[0])
                features_cluster = StandardScaler().
↪fit_transform(features_cluster)

                # 使用 DBSCAN 聚类
                dbscan = DBSCAN(eps=0.5, min_samples=2)
                group['Cluster'] = dbscan.fit_predict(features_cluster)

                # 为每个聚类生成唯一标识
                for cluster in group['Cluster'].unique():
                    cluster_group = group[group['Cluster'] == cluster]
                    for _, cluster_row in cluster_group.iterrows():
                        new_feature = f"{feature}_{cluster}"
                        processed_data.append({
                            'Name': cluster_row['Name'],
                            'Sex': cluster_row['Sex'],
                            'Team': cluster_row['Team'],
                            'NOC': cluster_row['NOC'],
                            'Year': cluster_row['Year'],
                            'Sport': cluster_row['Sport'],
                            'Event': cluster_row['Event'],
                            'Feature': new_feature
                        })
                        unique_feature_count += 1
                break  # 已经处理完当前分组，跳出循环

        # 如果时间跨度在阈值内，认为是同一个运动员
        new_feature = f"{feature}_{unique_feature_count}"
        processed_data.append({
            'Name': row['Name'],
            'Sex': row['Sex'],
            'Team': row['Team'],
            'NOC': row['NOC'],
            'Year': row['Year'],
            'Sport': row['Sport'],
```

```
            'Event': row['Event'],
            'Feature': new_feature
        })

        # 更新变量
        last_year = current_year

# 将处理后的数据转换为 DataFrame
processed_df = pd.DataFrame(processed_data)

# 显示处理后的数据
print("\n处理后的数据: ")
print(processed_df[['Feature', 'Sport', 'Event', 'Year']].head())

# 保存处理后的数据到新的 CSV 文件
output_file_path = 'athlete_years_processed.csv'
processed_df.to_csv(output_file_path, index=False)
print(f"\n处理后的数据已保存到 {output_file_path}")
```

原始数据的前几行:

```
            Name Sex  NOC          Team  Year          Sport  \
0   (jr) Larocca   M  ARG     Argentina  2024     Equestrian
1  . Chadalavada   F  IND         India  2020        Fencing
2         . Deni   M  INA     Indonesia  2020  Weightlifting
3            671   F  CHN         China  2024       Breaking
4       A Alayed   F  KSA  Saudi Arabia  2024       Swimming

                      Event               Feature
0        Jumping Individual   (jr) Larocca, M, ARG
1  Women's Sabre Individual  . Chadalavada, F, IND
2               Men's 67kg         . Deni, M, INA
3                  B-Girls            671, F, CHN
4    Women's 200m Freestyle       A Alayed, F, KSA
```

处理后的数据:

```
                  Feature          Sport                      Event  Year
0   (jr) Larocca, M, ARG_0     Equestrian        Jumping Individual  2024
```

```
1   . Chadalavada, F, IND_0        Fencing  Women's Sabre Individual  2020
2         . Deni, M, INA_0  Weightlifting                Men's 67kg  2020
3            671, F, CHN_0       Breaking                   B-Girls  2024
4       A Alayed, F, KSA_0       Swimming    Women's 200m Freestyle  2024
```

处理后的数据已保存到 athlete_years_processed.csv

2.3.3 统计连续参加奥运会的年数与对应人数

[14]:
```
# 读取 CSV 文件
file_path = 'Generated\\athlete_years_processed.csv'  # 替换为你的文件路径
athlete_years = pd.read_csv(file_path)
```

[15]:
```
# 计算连续参加的届数
def count_consecutive_years(group):
    years = group['Year'].sort_values().values
    consecutive_year = []
    current_count = 1
    for i in range(1, len(years)):
        if years[i] - years[i - 1] <= 6 :
            if years[i] - years[i - 1] >= 3:
                current_count += 1
        else:
            if current_count > 10:
                print(group)
            consecutive_year.append(current_count)
            current_count = 1
    consecutive_year.append(current_count)
    return pd.Series(consecutive_year)

# 应用函数计算每个运动员的连续届数
consecutive_years = athlete_years.groupby('Feature').
 ↪apply(count_consecutive_years, include_groups=False).explode().reset_index()
consecutive_years.columns = ['Feature', 'level_0',  'Consecutive_Years']  # 修正
列名
consecutive_years = consecutive_years.drop(columns=['level_0'])  # 删除不必要的
列
```

```
# 统计每个连续届数的人数
consecutive_years_count = consecutive_years['Consecutive_Years'].value_counts().
 ↪reset_index()
consecutive_years_count.columns = ['Consecutive_Years', 'Count']

# 输出结果
print("连续参加奥运会的届数与对应人次：")
print(consecutive_years_count)

# 保存为新的 CSV 文件
output_path = 'Generated\\consecutive_years_count.csv'
consecutive_years_count.to_csv(output_path, index=False, encoding='utf-8')
print(f"统计结果已保存到 {output_path}")
```

连续参加奥运会的届数与对应人次：

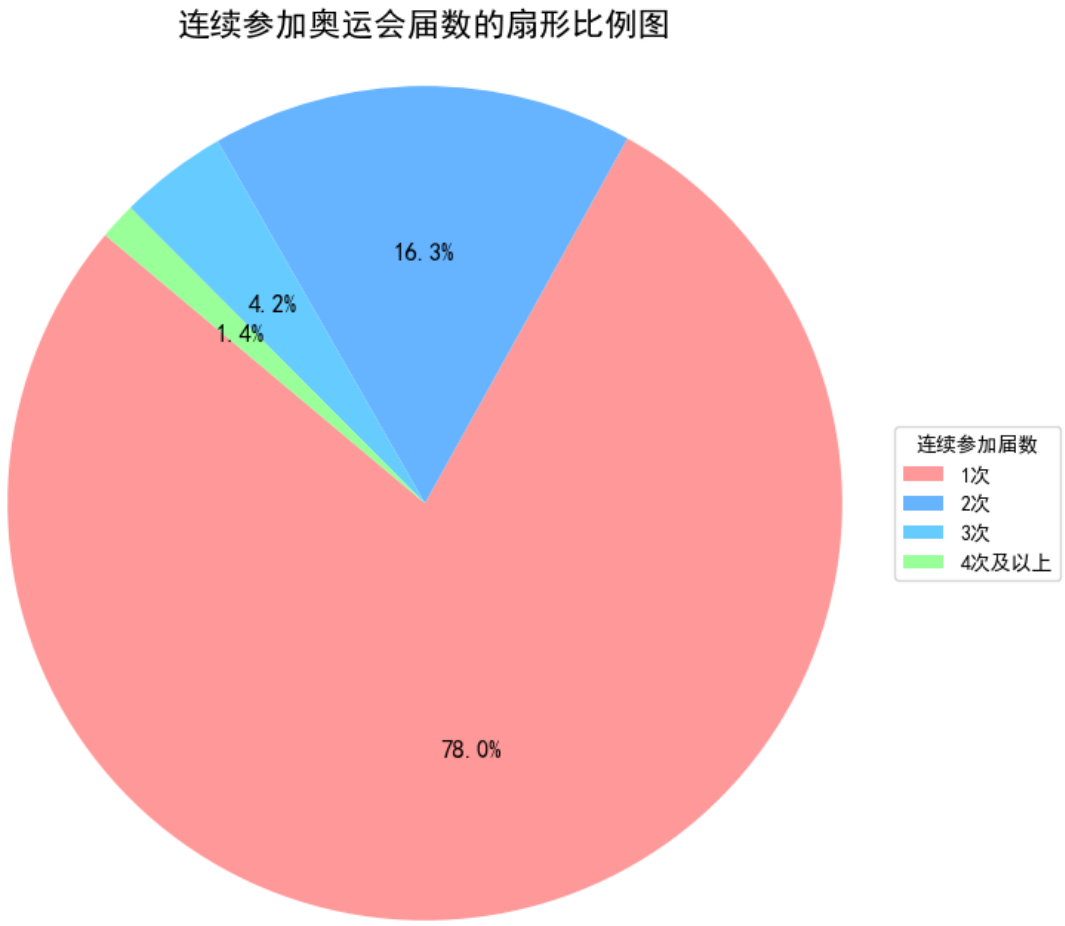
	Consecutive_Years	Count
0	1	110747
1	2	23175
2	3	5959
3	4	1543
4	5	367
5	6	79
6	7	18
7	8	4
8	9	1

统计结果已保存到 Generated\consecutive_years_count.csv

数据可视化

[16]:
```
# 导入必要的库
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties

# 设置支持中文的字体
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体字体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题
```

```python
# 读取数据
data = pd.read_csv("Generated/consecutive_years_count.csv")

# 定义大致届数区间
bins = [0, 2, 3, 4, 14]
labels = ['1 次', '2 次', '3 次', '4 次及以上']

# 将数据分组到区间
data['Group'] = pd.cut(data['Consecutive_Years'], bins=bins, labels=labels,␣
 ↪right=False)

# 计算每个区间的总人次，显式设置 observed=True
grouped_data = data.groupby('Group', observed=True)['Count'].sum().reset_index()

# 准备绘图数据
labels = grouped_data['Group']
sizes = grouped_data['Count']
colors = ['#ff9999', '#66b3ff', '#66ccff', '#99ff99']  # 颜色列表

# 绘制饼图
plt.figure(figsize=(8, 8))
wedges, texts, autotexts = plt.pie(sizes, colors=colors, autopct='%1.1f%%',␣
 ↪startangle=140, textprops={'fontsize': 12})

# 添加图例（色块 + 标签），放置在右侧
plt.legend(wedges, labels, title="连续参加届数", loc="center left",␣
 ↪bbox_to_anchor=(1, 0, 0.5, 1))

plt.title('连续参加奥运会届数的扇形比例图', fontsize=16)
plt.axis('equal')  # 确保饼图是圆形
plt.show()
```



[17]:
```
# 保存组别与对应比例
group_percentages = []
for label, autotext in zip(labels, autotexts):
    # 获取百分比文本并去掉百分号，转换为浮点数
    percentage = float(autotext.get_text().strip('%'))
    group_percentages.append((label, percentage))

# 打印结果
print("组别与对应比例：")
for group, percentage in group_percentages:
    print(f"{group}: {percentage:.1f}%")
```

组别与对应比例：

1 次：78.0%

2 次：16.3%

3 次：4.2%

4 次及以上：1.4%

2.3.4 统计运动员参加奥运会的时间跨度

[18]:
```
# 读取 CSV 文件
file_path = 'Generated\\athlete_years_processed.csv'  # 替换为你的文件路径
athlete_years = pd.read_csv(file_path)
```

[19]:
```
# 计算每个运动员的第一次和最后一次参赛年份
def calculate_year_gap(group):
    years = group['Year'].values
    min_n = 2032
    max_n = 1896
    for i in years:
        if i < min_n:
            min_n = i
        if i > max_n:
            max_n = i
    if len(years) > 0:
        if max_n - min_n + 1 > 60:
            #print(group)
            return 1
        return max_n - min_n + 1
    else:
        return 0

# 应用函数计算每个运动员的间隔年数
athlete_gaps = athlete_years.groupby('Feature').apply(calculate_year_gap,␣
 ↪include_groups=False).reset_index()
athlete_gaps.columns = ['Feature', 'Year_Gap']

# 统计每个间隔年数的人数
gap_counts = athlete_gaps['Year_Gap'].value_counts().reset_index()
gap_counts.columns = ['Year_Gap', 'Count']
```

```python
# 按 Year_Gap 排序
gap_counts = gap_counts.sort_values(by='Year_Gap')

# 输出结果
print("运动员第一次参加奥运会和最后一次参加奥运会之间的间隔年数：")
print(gap_counts)

# 保存为新的 CSV 文件
output_path = 'Generated\\athlete_year_gaps.csv'
gap_counts.to_csv(output_path, index=False, encoding='utf-8')
print(f"统计结果已保存到 {output_path}")
```

运动员第一次参加奥运会和最后一次参加奥运会之间的间隔年数：

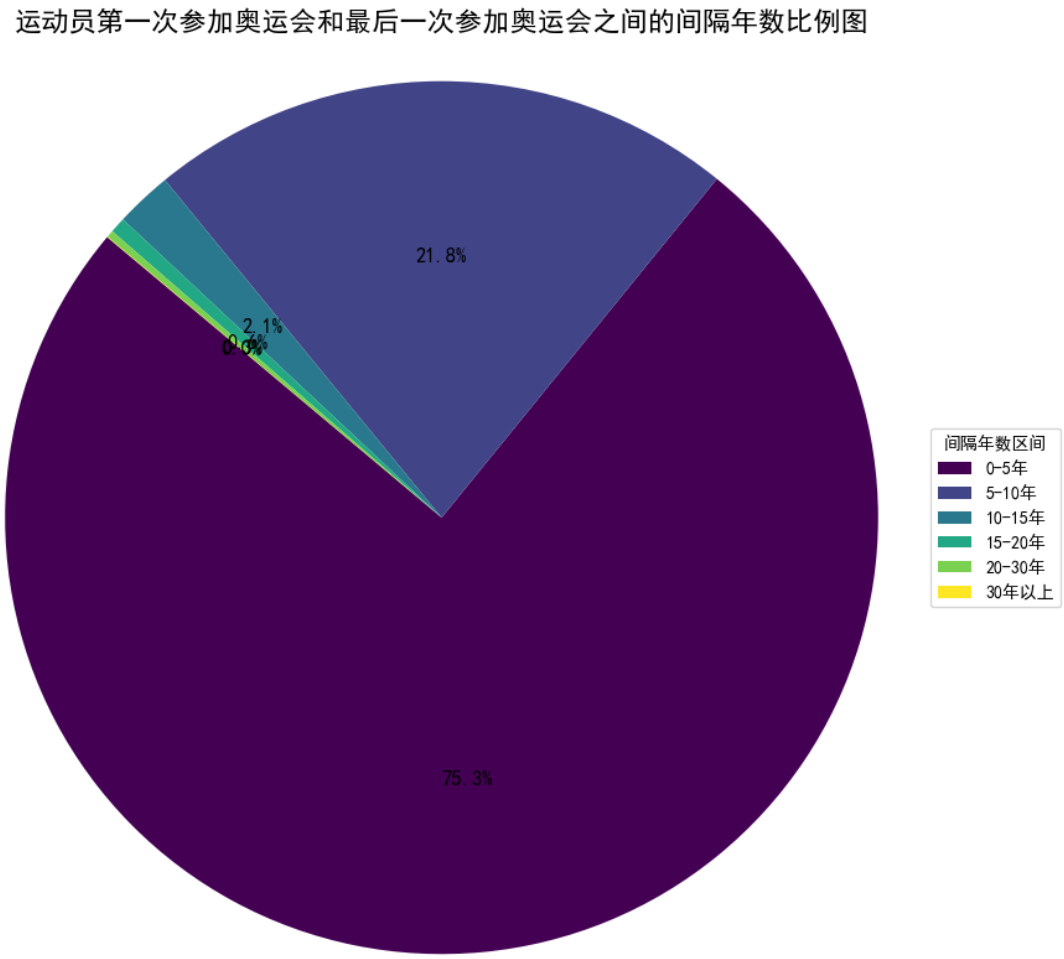
	Year_Gap	Count
0	1	103205
6	3	98
1	5	21831
8	7	77
2	9	8000
11	11	12
3	13	2872
12	15	12
4	17	781
13	19	5
5	21	233
16	23	2
7	25	86
18	27	1
9	29	37
10	33	16
14	37	3
15	41	3
17	45	2
20	49	1
19	53	1

统计结果已保存到 Generated\athlete_year_gaps.csv

数据可视化

[20]:
```
# 导入必要的库
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 设置支持中文的字体
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体字体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 读取数据
data = pd.read_csv("Generated\\athlete_year_gaps.csv")

# 定义大致间隔年数区间
bins = [0, 5, 10, 15, 20, 30, 120]  # 区间划分：0-5 年，5-10 年，10-15 年，15-20 年，20-30 年，30 年以上
labels = ['0-5 年', '5-10 年', '10-15 年', '15-20 年', '20-30 年', '30 年以上']

# 将数据分组到区间
data['Group'] = pd.cut(data['Year_Gap'], bins=bins, labels=labels, right=False)

# 计算每个区间的总人次
grouped_data = data.groupby('Group', observed=True)['Count'].sum().reset_index()

# 准备绘图数据
labels = grouped_data['Group']
sizes = grouped_data['Count']
colors = plt.cm.viridis(np.linspace(0, 1, len(labels)))  # 使用颜色映射生成颜色列表

# 绘制饼图
plt.figure(figsize=(10, 10))
wedges, texts, autotexts = plt.pie(sizes, colors=colors, autopct='%1.1f%%',␣
 ↪startangle=140, textprops={'fontsize': 12})

# 添加图例（色块 + 标签），放置在右侧
```

```
plt.legend(wedges, labels, title="间隔年数区间", loc="center left",␣
 ↪bbox_to_anchor=(1, 0, 0.5, 1))

plt.title('运动员第一次参加奥运会和最后一次参加奥运会之间的间隔年数比例图',␣
 ↪fontsize=16)
plt.axis('equal')  # 确保饼图是圆形
plt.show()
```



‘ ’’ 根据扇形图，对于运动员连续参加比赛，只考虑连续参加 2-3 届的运动员的连续性影响，其余影响可以忽略不计。’ ’ ’

参加时间跨度为 0-15 年的运动员中连续参加的比例

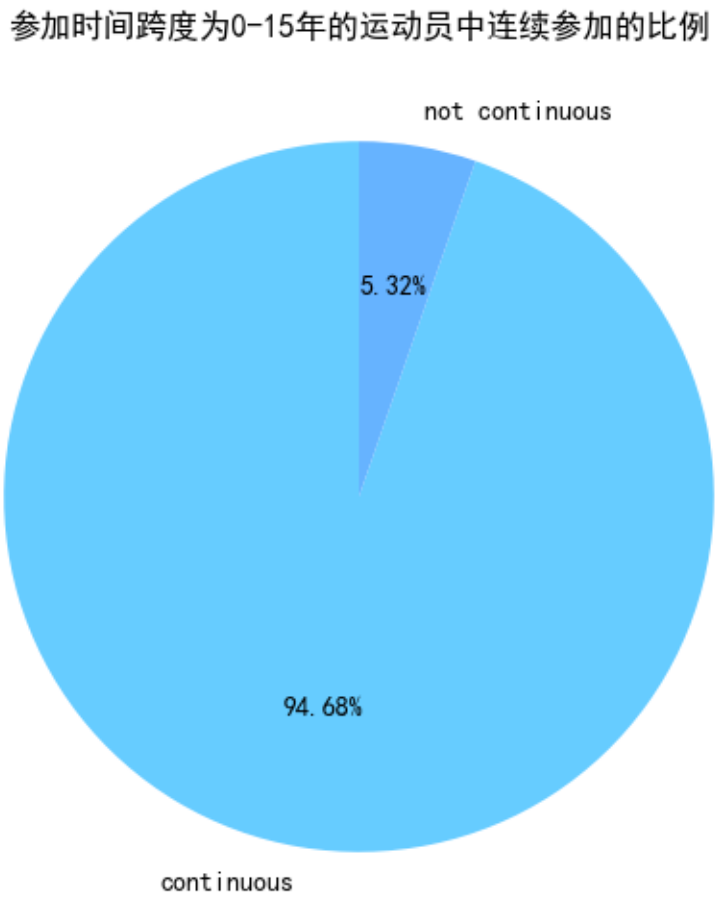
[21]:
```
# 合并时间跨度和连续届数数据
athlete_gaps.to_csv('Generated\\athlete_gaps.csv')
consecutive_years.to_csv('Generated\\consecutive_years.csv')
merged_data = athlete_gaps.merge(consecutive_years, on='Feature')

# 筛选出时间跨度为 1-15 年的运动员
filtered_data = merged_data[(merged_data['Year_Gap'] >= 1) &␣
 ↪(merged_data['Year_Gap'] <= 15)]

# 统计连续参加的比例
total_count = filtered_data.shape[0]
consecutive_count = filtered_data[filtered_data['Year_Gap'] <=␣
 ↪filtered_data['Consecutive_Years']*4].shape[0]
consecutive_ratio = consecutive_count / total_count if total_count > 0 else 0

# 输出结果
print(f"时间跨度为 1-15 年的运动员中，连续参加的比例为: {consecutive_ratio:.2%}")

# 保存结果到 CSV 文件
output_path = 'Generated\\consecutive_ratio.csv'
filtered_data.to_csv(output_path, index=False, encoding='utf-8')
print(f"统计结果已保存到 {output_path}")
```

```
时间跨度为 1-15 年的运动员中，连续参加的比例为: 94.68%
统计结果已保存到 Generated\consecutive_ratio.csv
```

数据可视化

[28]:
```
# 导入必要的库
import matplotlib.pyplot as plt

# 设置支持中文的字体
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体字体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号显示问题

# 数据
percentages = [consecutive_ratio*100, (1-consecutive_ratio)*100]  # 一个百分数和
剩余部分
```

```
labels = ['continuous', 'not continuous']  # 标签
colors = ['#66ccff', '#66b3ff']  # 颜色

# 绘制饼图
plt.figure(figsize=(6, 6))  # 设置图形大小
plt.pie(percentages, labels=labels, colors=colors, autopct='%1.2f%%',␣
 ↪startangle=90)
# autopct='%1.2f%%' 表示在每个扇形上显示百分比，格式为 2 位小数
# startangle=90 表示从 90 度（即正上方）开始绘制

# 添加标题
plt.title('参加时间跨度为 0-15 年的运动员中连续参加的比例')

# 显示图形
plt.show()
```



结论

- ‘我们可以发现，参加奥运会时间跨度 0-15 年中绝大部分运动员都是连续参加的’
- ‘而且我们前面发现，绝大部分的运动员的时间跨度在 0-15 年之间，连续参加届数在 1-3 届’
- ‘而且我们知道，0-15 之间只能连续参加 1-3 次奥运会’
- ‘我们因此可以得出结论，绝大部分奥运会运动员连续参加了 1-3 次奥运会’
- ‘所以我们可以得出结论，考虑运动员连续参加比赛对奖牌的影响只需要考虑连续参加 2-3 次的情况’

[31]:
```
first_percentage = group_percentages[1][1]/
 ↪(group_percentages[0][1]+group_percentages[1][1]+group_percentages[2][1]+group_percentages[
print(f'一个参加了一次奥运会的运动员参加下一次奥运会的可能为{first_percentage : .
 ↪2f}' + '%')
second_percentage = group_percentages[2][1]/
 ↪(group_percentages[1][1]+group_percentages[2][1]+group_percentages[3][1])*100
print(f'一个参加了两次奥运会的运动员参加下一次奥运会的可能为{second_percentage : .
 ↪2f}' + '%')
third_percentage = group_percentages[3][1]/
 ↪(group_percentages[2][1]+group_percentages[3][1])*100
print(f'一个参加了三次奥运会的运动员参加下一次奥运会的可能为{third_percentage : .
 ↪2f}' + '%')
athlete_join_willing = {1 : first_percentage, 2 : second_percentage, 3 :␣
 ↪third_percentage}
```

```
一个参加了一次奥运会的运动员参加下一次奥运会的可能为 16.32%
一个参加了两次奥运会的运动员参加下一次奥运会的可能为 19.18%
一个参加了三次奥运会的运动员参加下一次奥运会的可能为 25.00%
```

3 构建模型