项目说明文档

1、数据的抓取

数据的主要获取途径:

[1896-2016奥运选手资料]

[2021年东京奥运数据]

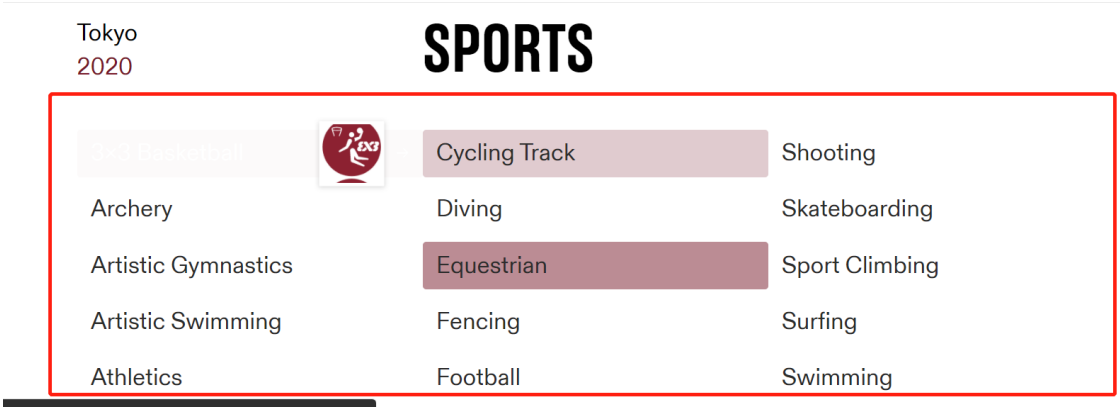
[奥运项目汇总]

关于参赛队伍的数据，参赛人数，男女人数比例都可以从上面两个数据集获取到，这里就不过多描述了

然后这里因为缺少了2020年奥运项目的数量，所以我们需要通过奥运国际网站去抓取项目的汇总

首先我们先进入东京奥运会的官网:https://olympics.com/en/olympic-games/tokyo-2020/results

然后去获取这些对应的内容便是奥运的全部项目



因为考虑到可能会遇到反爬策略，所以我们需要构建一个请求头用来防止被反爬

```
headers = {
        'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.102 Safari/537.36'
    }
```

接着我们首先先去用requests去获取页面的内容并且检查当前网页是否能正常被请求

```
def get_parse(url):
    html = requests.get(url, headers=headers)
    if html.status_code == 200:
        xianmu_number(html)
    else:
        print(html.status_code)
```

接着我们获取的时候要用正则表达式去定位，因为这里加了一些反爬干扰，所以我们用xpath定位是失败的，只能用正则表达式去从根源上进行定位

```
def xianmu_number(html):
    content = html.text
    project = re.compile('"sportDisciplineId":".*?","title":"(.*?)"}.*?,')
    projects = project.findall(content)
    df = pd.DataFrame()
    df['project'] = projects
    df.to_excel('./data/2020/olympics_project.xlsx')
```

获取到内容之后，我们用pandas库把数据保存起来到我们的指定路径下

2、数据清洗以及数据可视化展示

在获取到所有数据后，开始我们的数据清洗步骤，首先先查看哪些是我们需要的数据

2016之前的数据用的是这张表

```
[3]: olympics_2016 = pd.read_csv('./data/1896-2016/athlete_events.csv')
olympics_2016.head()
```

ut[3]:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

2020的数据用的是这张表

```
]: olympics_2020_team = pd.read_excel('./data/2020/Athletes.xlsx')
olympics_2020_team.head()
```

]:

	Name	NOC	Discipline
0	AALERUD Katrine	Norway	Cycling Road
1	ABAD Nestor	Spain	Artistic Gymnastics
2	ABAGNALE Giovanni	Italy	Rowing
3	ABALDE Alberto	Spain	Basketball
4	ABALDE Tamara	Spain	Basketball

首先我们使用groupby对数据进行分组，找出每一年对应的数据

分别找出它们每年的比赛队伍数量，参赛人数，男女之比，和项目数量

```
: data1 = olympics_2016.groupby('Games').apply(teams_number)
```

```
[10]: data3 = olympics_2016.groupby('Games').apply(people_number)
```

```
]: data4 = olympics_2016.groupby('Games').apply(sex_number)
```

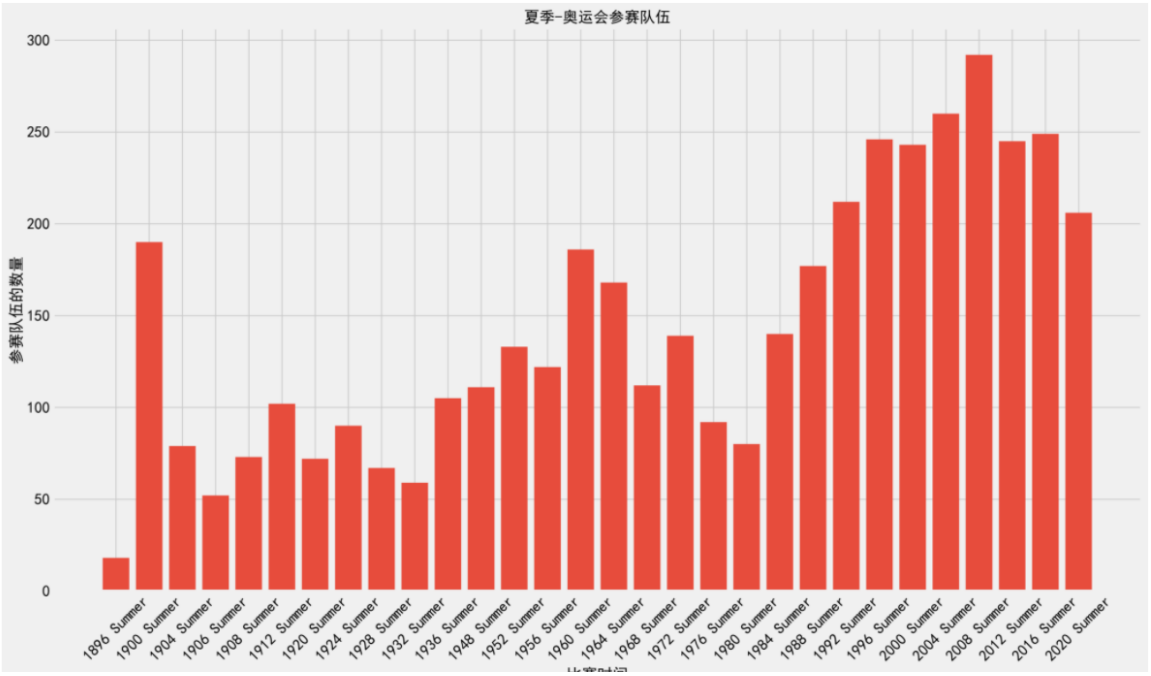
```
17]: data5 = olympics_2016.groupby('Games').apply(project_number)
```

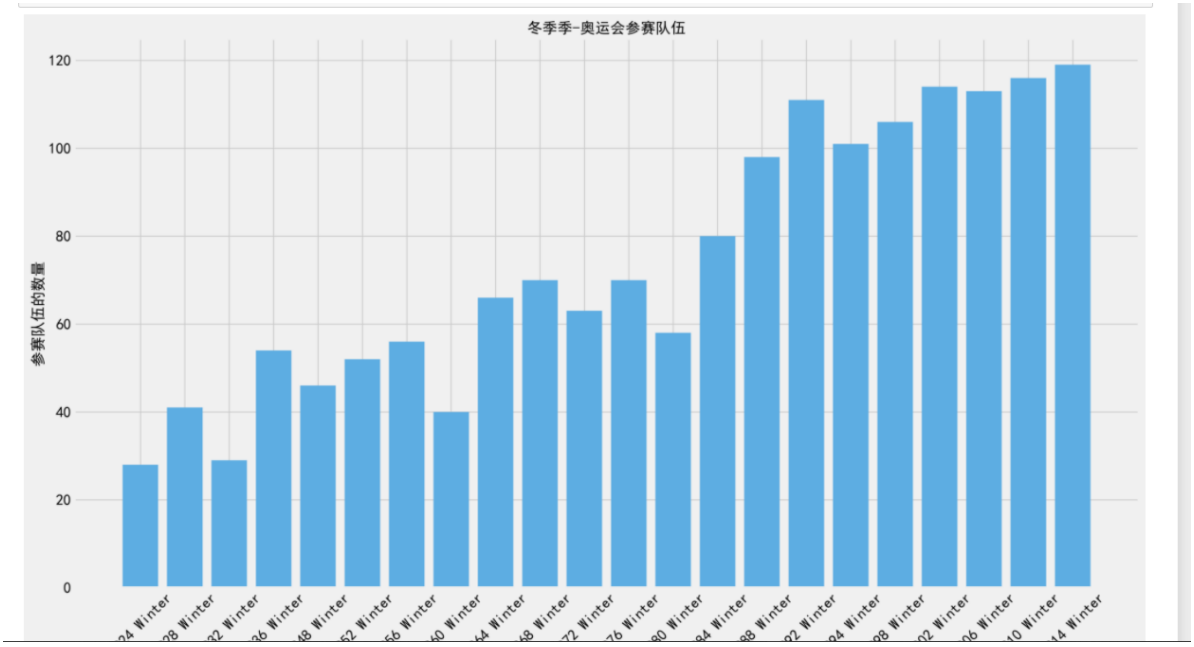
把这些数据全部获取好之后

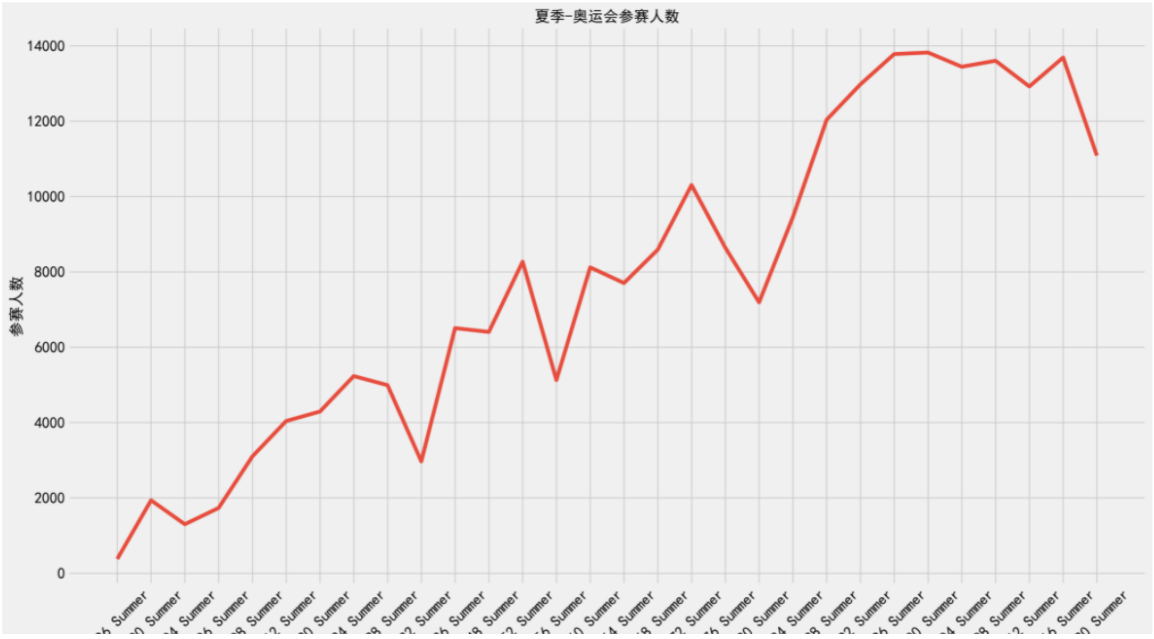
我们去创建一些列表，先把它们冬奥运和夏奥运区分开

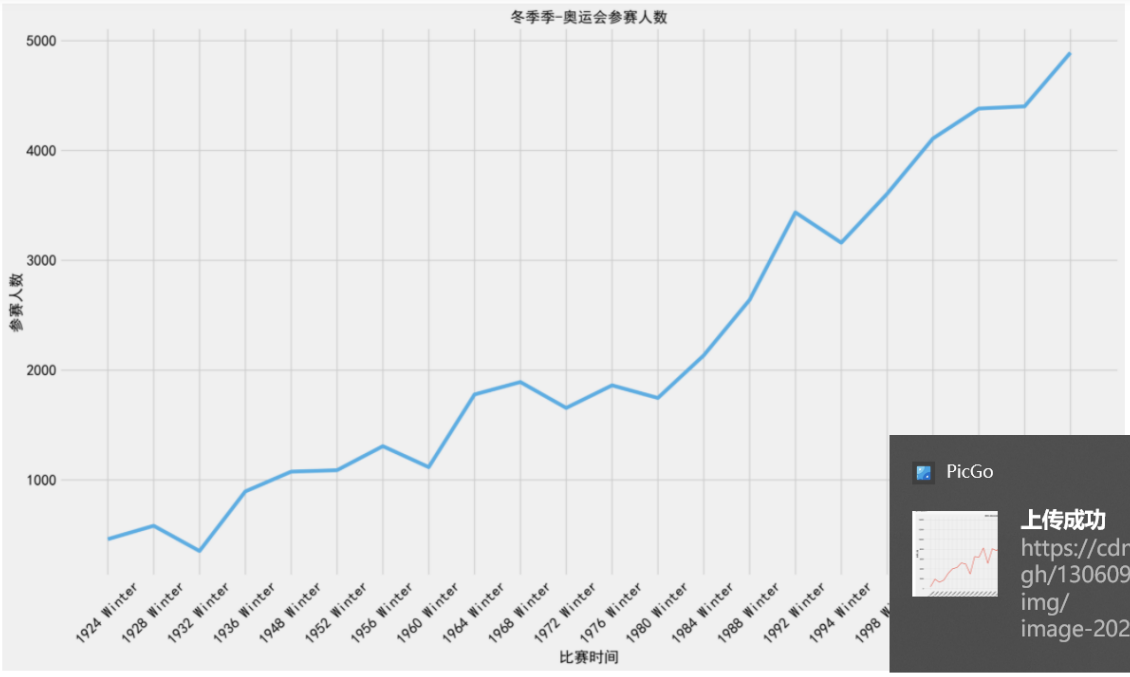
```
x_summer = []
y_summer = []
x_winter = []
y_winter = []

for j,k in zip(x_data,y_data):
    if 'Summer' in j:
        x_summer.append(j)
        y_summer.append(k)
    if 'Winter' in j:
        x_winter.append(j)
        y_winter.append(k)
```

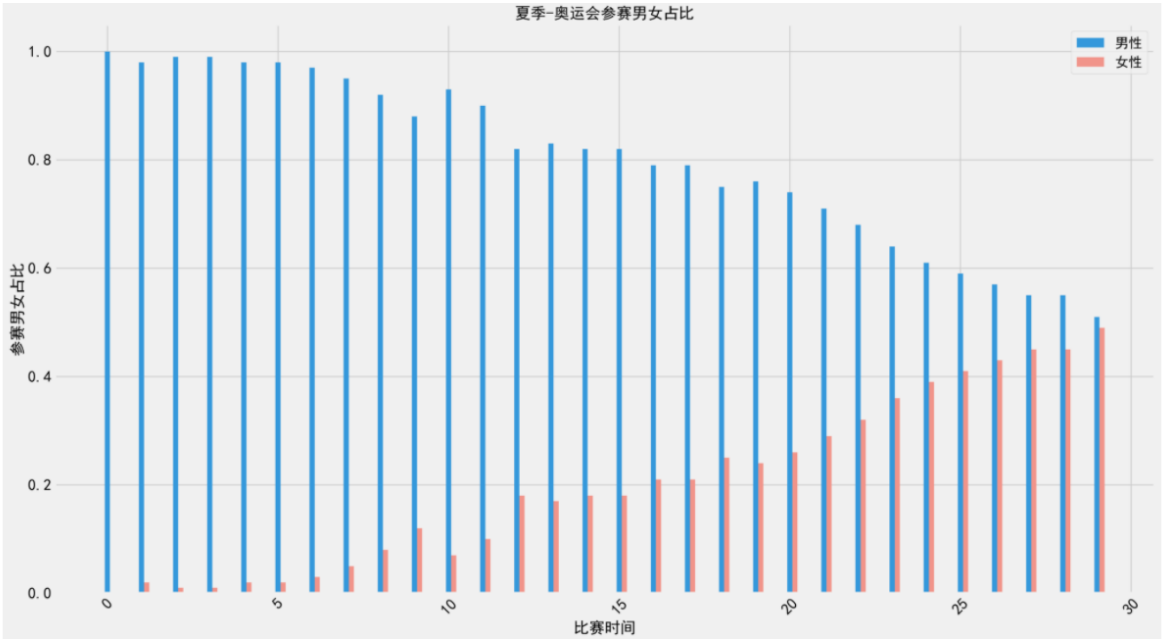
并且用可视化的方式展示出来

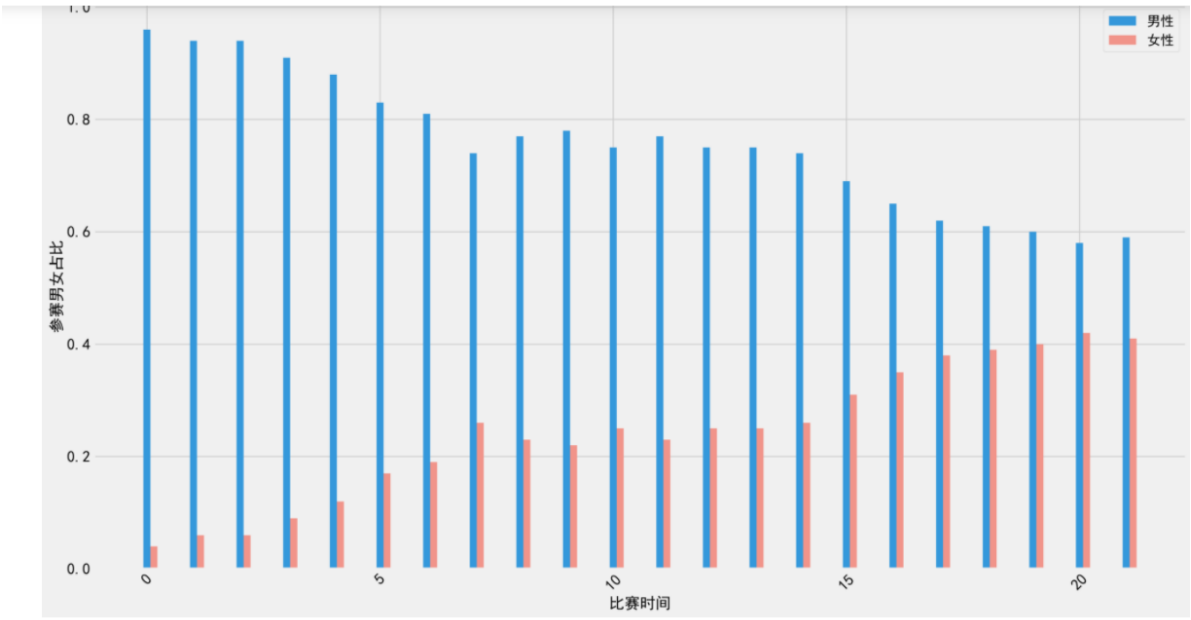


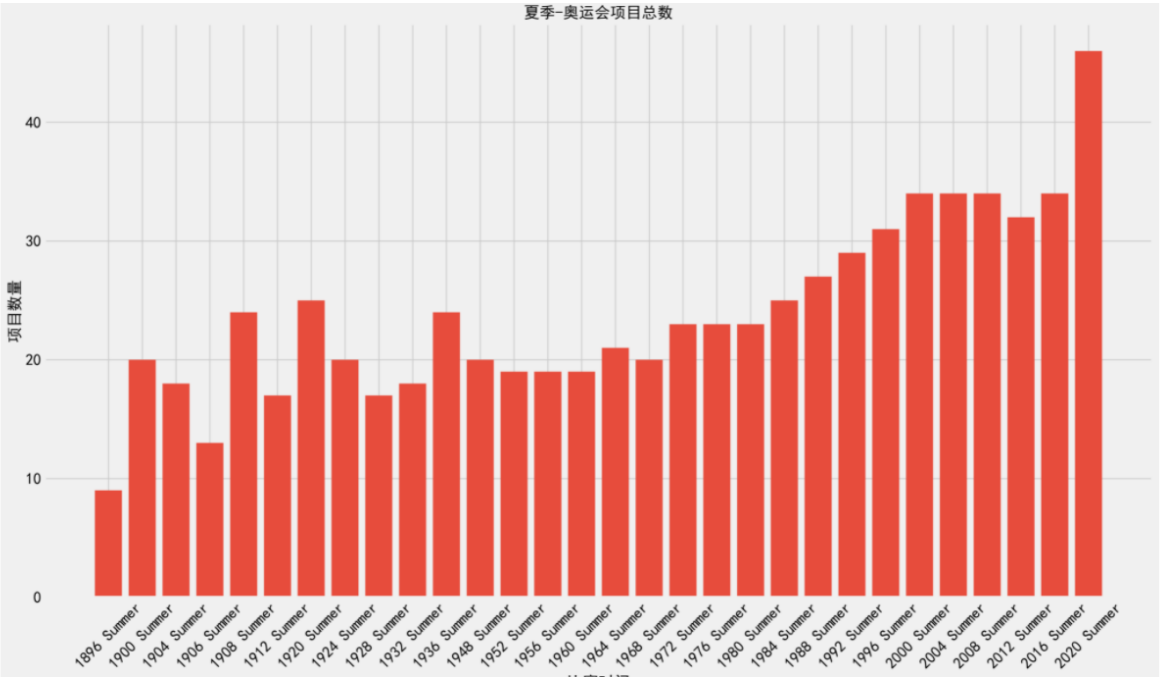


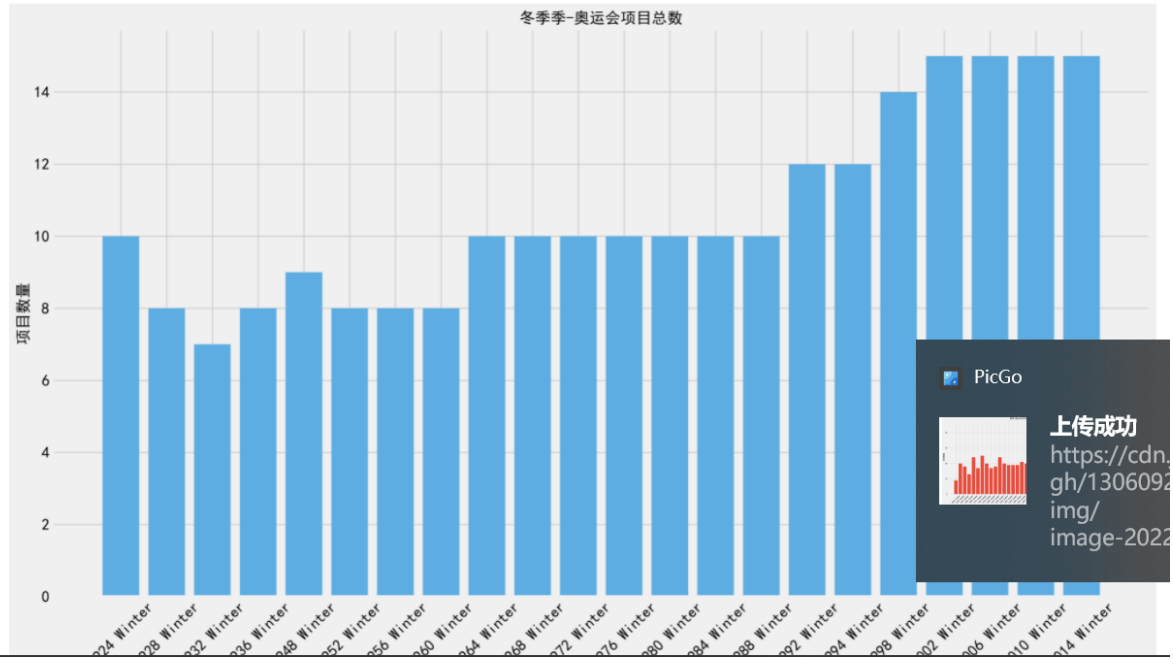












当我们把全部想要的数据可视化之后，我们再来对一些数据进行数据分析

这里我们通过pct_change函数去计算参赛队伍，参赛人数，项目每年增长率

然后再通过这些增长率去计算下一次奥运预估增长的数量

这是夏季的

```
df_summer['team_addional'] = df_summer.teams.pct_change(periods=1,fill_method='pad')
df_summer['team_people'] = df_summer.people.pct_change(periods=1,fill_method='pad')
df_summer['team_project'] = df_summer.project.pct_change(periods=1,fill_method='pad')
teams_2024 = int(df_summer['teams'][29]) + int(int(df_summer['teams'][29]) *  float(df_summer['team_addional'][29]))
people_2024 = int(df_summer['people'][29]) + int(int(df_summer['people'][29]) *  float(df_summer['team_people'][29]))
project_2024 = int(df_summer['project'][29]) + int(int(df_summer['project'][29]) *  float(df_summer['team_project'][29]))
df_2024_summer = pd.DataFrame([['2024 Summer',teams_2024,people_2024,np.NaN,np.NaN,project_2024,'Summer',2024]], columns=['time','t
df_summer = df_summer.append(df_2024_summer)
df_summer
```

这是冬季的

```
df_winter['team_addional'] = df_winter.teams.pct_change(periods=1,fill_method='pad')
df_winter['team_people'] = df_winter.people.pct_change(periods=1,fill_method='pad')
df_winter['team_project'] = df_winter.project.pct_change(periods=1,fill_method='pad')

teams_2018 = int(df_winter['teams'][21]) + int(int(df_winter['teams'][21]) *  float(df_winter['team_addional'][21]))
people_2018 = int(df_winter['people'][21]) + int(int(df_winter['people'][21]) *  float(df_winter['team_people'][21]))
project_2018 = int(df_winter['project'][21]) + int(int(df_winter['project'][21]) *  float(df_winter['team_project'][21]))
df_2018_winter = pd.DataFrame([['2018 Winter',teams_2018,people_2018,np.NaN,np.NaN,project_2018,'Winter',2018]], columns=['time','te
df_winter = df_winter.append(df_2018_winter)

df_winter
```

至于空余的部分我们采用fillna函数进行空值填补，fillna是用上面的内容替换空余值，因为最前面的一行，增长率是为0的，所以我们把最上面的一行用0代替

```
df_summer.reset_index(drop=True, inplace=True)
df_winter.reset_index(drop=True, inplace=True)
df_summer = df_summer.fillna(method='ffill')
df_winter = df_winter.fillna(method='ffill')
df_winter = df_winter.replace(np.NAN,0)
df_summer = df_summer.replace(np.NAN,0)
```

3、在整个数据查看完毕之后，我们再来查看我们的国家夏季奥运的表现

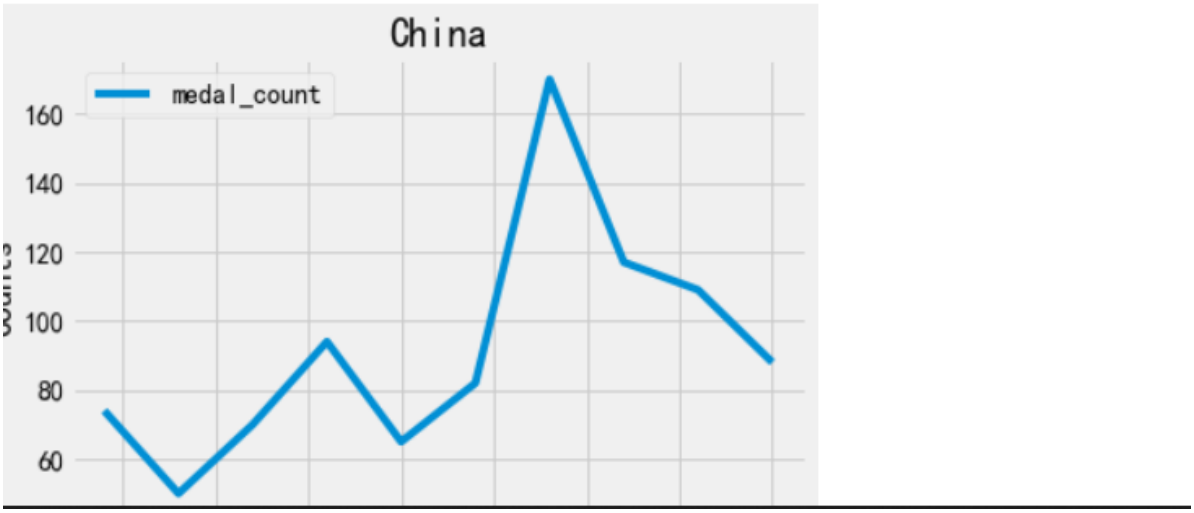
首先我们根据全部的数据

```
df_summer = olympics_2016[olympics_2016.Season=="Summer"]
athlete_winners = df_summer[df_summer.Medal.notnull()]
winner_team = athlete_winners.groupby(['Team','Year']).size().to_frame('medal_count').reset_index().sort_values('Year',ascend
winner_team = winner_team[winner_team.Team == "China"]
df = pd.DataFrame([['China',2020,88]], columns=['Team','Year','medal_count'])
```

去定位中国，夏季奥运，每次获取的奖牌总数

并且将数据用可视化的方式展示出来

```
  winner_team = winner_team.sort_values('Year',ascending=False)
  ax = winner_team.plot(x='Year',y='medal_count')
  ax.set_title("China")
  ax.set_xlabel('Year')
  ax.set_ylabel('Counts')
  plt.show()
  return winner_team
nner_team =  DrawAndCountCountry()
```



并且我们来分析一下这个图为什么是这样的

从上面的图形可以得知，从1986年开始，其实我们获取金牌都是越来越多的，最高峰是在2008年，之所以2008年获取金牌最多是因为
2008年那年我们作为主办方自然获取的金牌数量是最多的,排除2008年这个我们作为主办方的原因，其他的来，都是保持在一个增长的趋势，中途
可能有一些下跌的倾向，不过在下一次奥运会也会慢慢增长回来

4、数据异常检查，数据建模，时间序列模型，数量预测

首先我们将年份那一列转化为序列

```
]: winner_team = winner_team.sort_values('Year',ascending=True)
   c = pd.to_datetime(list(winner_team['Year']),format='%Y')
   winner_team.index = pd.to_datetime(c)
   winner_team
```

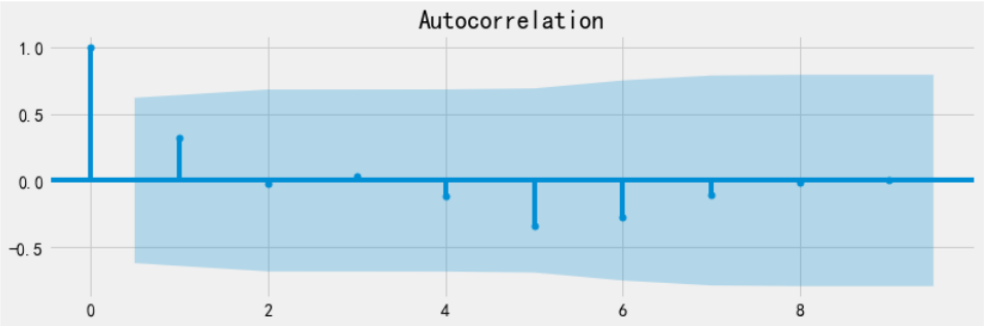
]:

	Team	Year	medal_count
1984-01-01	China	1984	74
1988-01-01	China	1988	50
1992-01-01	China	1992	70
1996-01-01	China	1996	94
2000-01-01	China	2000	65
2004-01-01	China	2004	82
2008-01-01	China	2008	170
2012-01-01	China	2012	117
2016-01-01	China	2016	109
2020-01-01	China	2020	88

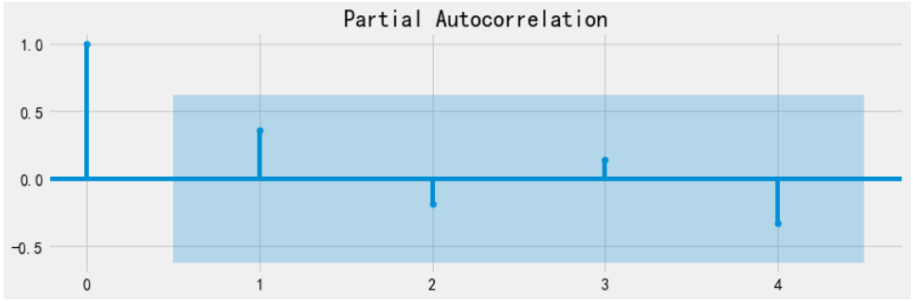
接着我们来查看arima模型中合适的p和q值

```
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_acf(data,lags=9,ax=ax1)
data
```

```
1984-01-01     74
1988-01-01     50
1992-01-01     70
1996-01-01     94
2000-01-01     65
2004-01-01     82
2008-01-01    170
2012-01-01    117
2016-01-01    109
2020-01-01     88
Name: medal_count, dtype: int64
```



81]:
```
#生成偏自相关图(PACF)
fig = plt.figure(figsize=(12,8))
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(data,lags=4,ax=ax2)
```



关于怎么查看上面这两个图形

可以参考这篇文章:如何根据自相关（ACF）图和偏自相关（PACF）图选择ARIMA模型的p、q值

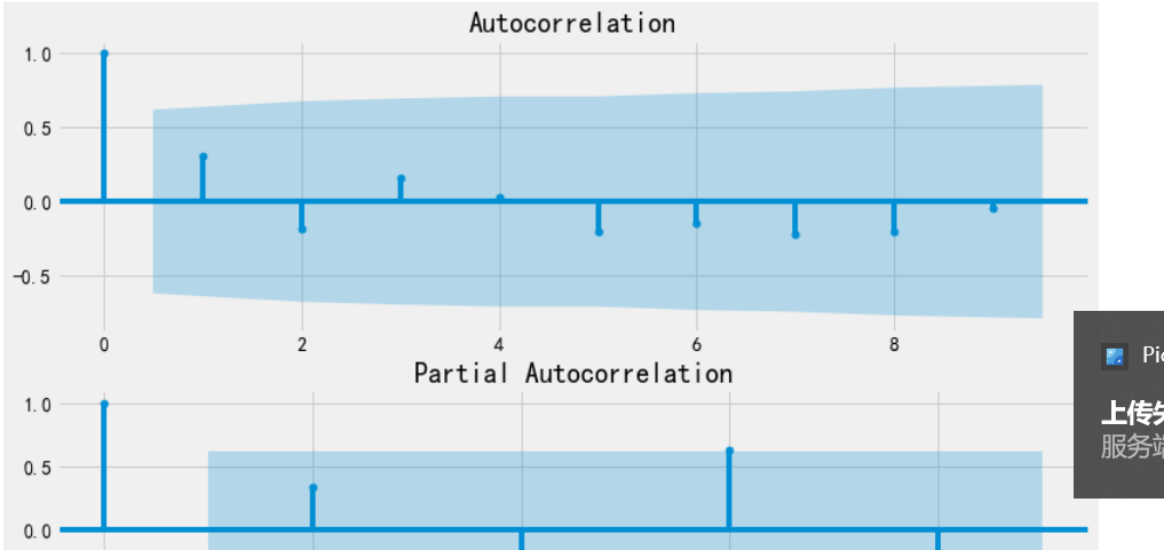
因为我也看不懂这篇文章，所以我这边直接使用代码的方式去查找最优的模型

```
from itertools import product
import warnings
from tqdm import tqdm
warnings.filterwarnings('ignore')
#设置参数范围
ps = range(0,10)
qs = range(0,10)
parameters = product(ps,qs)
parameters_list = list(parameters)


#寻找最优arma模型参数，即best_aic最小
results = []
#正无穷
best_aic = float('inf')

#去测试，寻找最优模型
for param in tqdm(parameters_list):
    try:
        model = ARMA(data,order=(param[0],param[1])).fit()
    except:
#         print('参数错误:',param)
        continue
    #一个比对的过程，选出最优的模型
    aic = model.aic
    if aic < best_aic:
        best_model = model
        best_aic = aic
        best_param = param
    results.append([param,model.aic])

#输出最优模型
result_table = pd.DataFrame(results)
result_table.columns = ['parameters','aic']
print('最优模型:',best_model.summary())
```

我们在这边设定了p,q值，然后让代码去把p,q值的所有可能都跑一边，从而获取最优的模型

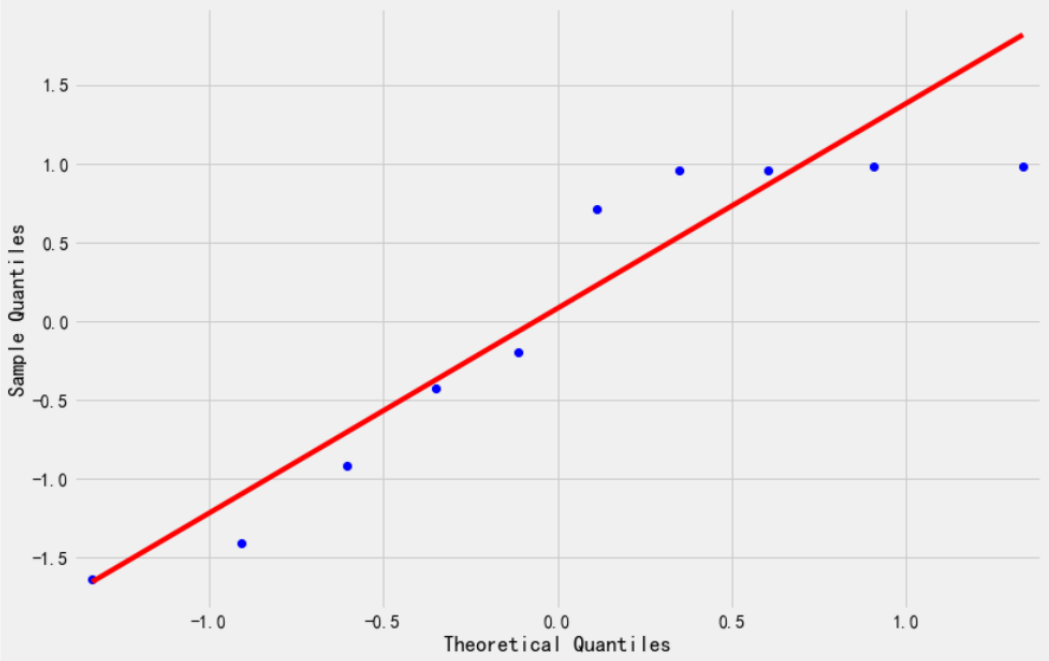
接着我们再去根据最优的模型重新去检索一下它的p,q图

```
#在选出模型检验之后，对arma模型所产生的残差做相关图
resid = best_model.resid
fig = plt.figure(figsize=(12,8))
ax1 = fig.add_subplot(211)
fig = sm.graphics.tsa.plot_acf(resid,lags=9,ax=ax1)
ax2 = fig.add_subplot(212)
fig = sm.graphics.tsa.plot_pacf(resid,lags=4,ax=ax2)
```



接着我们去查看我们数据是否符合正太分布，从图片上看，点都是接近红线的，也就说明我们是符合正态分布的

```
#基本符合正态分布
#生成正态图
from statsmodels.graphics.api import qqplot
fig = plt.figure(figsize=(12,8))
ax = fig.add_subplot(111)
fig = qqplot(resid, line='q', ax=ax, fit=True)
```



然后我们再去检查数据是否存在自相关性

因为最后一列都大于0.05，所以是不存在自相关性的

```
#用Ljung-box检验:检验结果就是看最后一列前十二行的检验概率
#prob值均大于0.05，所以残差序列不存在自相关性
r,q,p = sm.tsa.acf(resid.squeeze(),qstat=True)
data1 = np.c_[range(1,10),r[1:],q,p]
table = pd.DataFrame(data1,columns=['lag','AC','Q','Prob(>Q)'])
table.set_index('lag')
```

lag	AC	Q	Prob(>Q)
1.0	0.310012	1.281432	0.257633
2.0	-0.182384	1.780392	0.410575
3.0	0.157431	2.205271	0.530911
4.0	0.032092	2.225868	0.694296
5.0	-0.205469	3.239087	0.663180
6.0	-0.149659	3.911018	0.688717
7.0	-0.221431	5.872290	0.554738
8.0	-0.199600	8.262693	0.408239
9.0	-0.040992	8.464337	0.488112

```
delta = best_model.fittedvalues - data
```

我们来查看数据的拟合效果，值为0.76，拟合效果还是不错的

```
delta = best_model.fittedvalues - data
#它的值在0-1之间，越接近1，拟合效果越好
score = 1 - delta.var() / data.var()
score
```

```
0.7578589335306782
```

因为前面的检测数据的内容，都趋于理想状态，也就说明最后我们测试的效果较为精确，

最后我们来查看一下预测的效果图如何，这边采用最后的几个数据进行对比

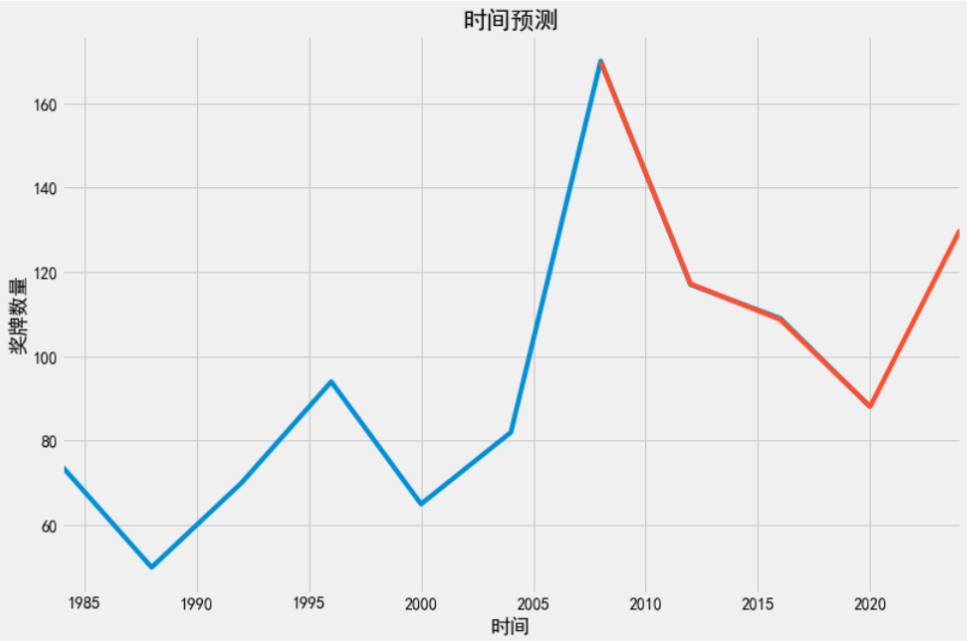
预测的结果是 170,117,109,88

而现实中的结果是170，117，109,88

基本处于完全吻合的状态，我们用可视化的方式展示更有说服力

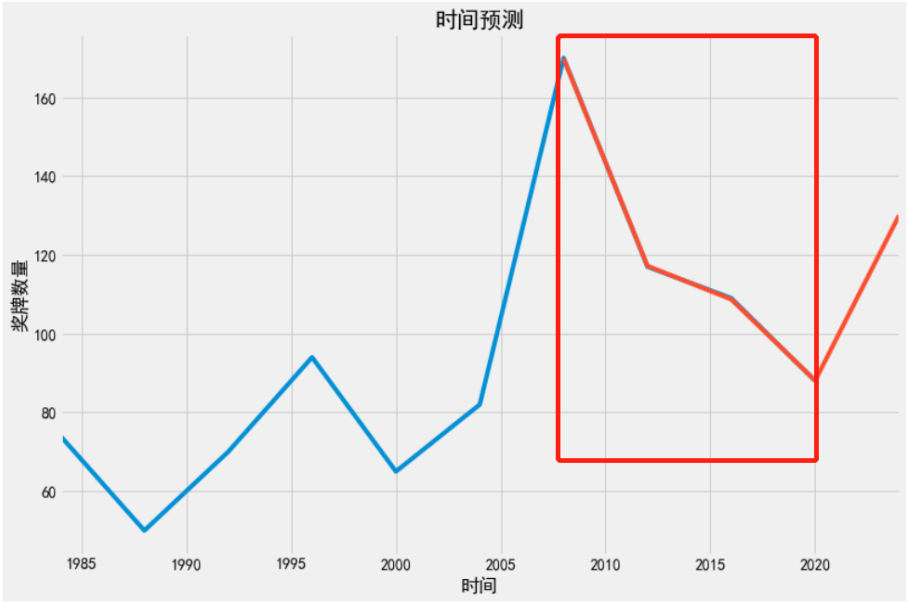
2]:
```
#导入时间库
from datetime import datetime
#转换成一维数值
data = pd.Series(data)
#预测后面几个月的趋势
predict_data = best_model.predict(start=6,end=10)
print(predict_data)
#定义画布大小
fig,ax = plt.subplots(figsize=(12,8))
#引入数据
ax = data.loc['1984-01-01':].plot(ax=ax)
predict_data.plot(ax=ax)
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus']=False
plt.title('时间预测')
plt.xlabel('时间')
plt.ylabel('奖牌数量')
plt.show()
```

```
2008-01-01    169.795495
2012-01-01    117.187107
2016-01-01    108.691660
2020-01-01     88.088492
2024-01-01    129.999638
Freq: 4AS-JAN, dtype: float64
```



[ ]:

```
Freq: 4AS-JAN, dtype: float64
```



这里面这一节里面是蓝色的，蓝色的线就是结果线，红色的线是预测线

这里不难看出，两条线是完全重叠的，也就是预测效果还行，预测的结果较为精准，因此可以作为结果值

最后我们预测2024年的中国将会在2024年获取130枚奖牌数量