Problem Chosen	2025 MCM/ICM Summary Sheet	Team Control Number
C		2515374

Foresting Greatness: Olympic Medal Predictions based on PRE Model

Summary

The Olympic Games, inspired by the ancient Greek Games, is the world's premier international multi-sport event. Held every four years, it features summer and winter editions, bringing together athletes from around the globe to compete in a wide range of sports. For every sports lover, the Stimulation of Olympic Medals is a very interesting topic, while it's crucial for countries to adjust their strategy. **In this paper, we will analyze the data of the Summer Olympic Games from 1896 to 2024, and use the data to simulate the results of the 2028 and 2032 Olympics using different methods from 'PRE' model mostly based on machine learning.**

First, we've examined, cleaned and transformed the raw data, setting the ice sports aside, combining various teams of one country, and mapping the countries which no longer exist to the current existing country. Then, we turn to machine learning, select the independent variables to use by **Correlation Coefficient Matrix** and **Principal Component Analysis**, split the data into training and testing parts, and use **eXtreme Gradient Boosting** methods to train features and target matrix. **Kolmogorov-Smirnov Test** is used to ensure the effectiveness of the model, while history data are reprocessed using feature importance and data in recent years are given higher weights to improve the accuracy of the model. In the end, we use the model to predict the results of the 2028 and 2032 Olympics, and the results are shown in multiple forms.

After that, we encode country labels, create feature dataset of each country,and filter out data for countries that won medals in 1896 to construct training and testing datasets of the feature of 'First Win Country'.**Random Forest** is used to train the model, and the results of the top ten countries most likely to win their first medal in 2028 are shown in the form of a bar chart.

By analyzing the data given, we find that some specific sports and events play a significant role in the medal tally of some specific countries, e.g: long distance race for Kenya. We calculate the **proportion** of medals from these events, further estimate their importance, and finally come to the result of the extent to which choosing these events impact countries performance in the medal list.

Finally, we combined methods used above and created a comprehensive prediction model called **'PRE'** model, naming after the primary methods we use. The model is specifically-tuned to calculate **Legendary Index** so as to detect **Great Couch Effect** using data of US gymnastics team coached by Bela Karolyi and Marta Karolyi. Lang Ping is successfully detected as a legendary coach, and we've found many more great teams and athletes, even controversial results(2024 Male Fencing,Italy) and decline of the great ranks(2024 Tennis,China).

Greatness is not born, but made. We hope that our model may help countries' Olympic committes to accommodate strategies and achieve better results in the future Olympic Games.

Keywords: Olympic Games; Data Analysis; Stimulation; Machine Learning

Contents

1 Introduction

1.1 Background and Literature Review

*T*he Olympic Games, often simply referred to as the Olympics, are the world's foremost international multi-sport events. With a history that dates back over 2,000 years to ancient Greece, the modern Olympics were revived in 1896 by Pierre de Coubertin.The Olympics are held every four years, with the Summer Olympics and the Winter Olympics alternating every two years. The Summer Olympics feature a vast array of sports, from athletics and swimming, which are considered the cornerstones of the Games, to more specialized sports like fencing, badminton, and gymnastics. Athletes from around the globe gather to compete at the highest level, showcasing their extraordinary skills, determination, and physical prowess.

The prediction of Olympic medal counts has long been a topic of interest among sports enthusiasts, statisticians, and researchers. Understanding how to forecast the number of medals a country or athlete might win is not only a matter of curiosity but also has implications for sports management, marketing, and national pride.

Traditional approaches are typically made closer to the start of an upcoming Olympic Games when information about the current athletes scheduled to compete becomes available. This approach allows for a more accurate assessment of a country's or athlete's medal prospects. For example, the virtual medal table forecast by Nielsen [1] provides a more real-time and data-driven prediction. By incorporating current athlete performance, injuries, and recent competition results, these modern models can better capture the dynamic nature of sports.

However, these data may be concealed and intentionally modified by countries' Olympic committees, which may produce misleading predictions.As a result, our model will be based on the historical data of the Olympic Games, which is more reliable and less likely to be manipulated. Research in this area has also explored the use of advanced statistical techniques. Machine learning algorithms, for instance, have been employed to analyze large datasets encompassing a wide range of variables related to athletes, sports, and countries. These algorithms can identify complex patterns and relationships that might not be apparent through traditional statistical methods.

In conclusion, the field of predicting Olympic medal counts has evolved significantly over the years. While historical contemporary methods still provide some basis for understanding trends, the focus has shifted towards historical data. The use of advanced statistical techniques and an increased awareness of external factors have improved the accuracy of these predictions. However, there is still room for further research, particularly in the use of machine learning methods and elements that can't be captured directly by historical data, such as the Great Coach Effect. As the Olympics continue to evolve, so too will the methods and models used to predict the medal counts, ensuring that this remains a vibrant and relevant area of study.

1.2 Restatement of the Problem

Considering the background, in this paper we are required to solve the following problems:

- **Task 1:** Develop a model for medal counts for each country, both **Gold** and **Total**, and use the model to predict various countries' performance in the 2028 Olympics. The model also includes estimates of the precision and measures of how well the model performs.

- **Task 2:** Develop a model for prediction of when a country will win its **first** medal in the Olympics, and the probability of winning it in the coming next LA Olympics. Also, we will evaluate this model.
- **Task 3:** Develop a model for estimation of the **"Great Coach"** effect, and further using this model to identify three countries suitable for imitating this strategy.
- **Task 4:** Calculating the **"Great Athlete"** effect, referring to the phenomenon that some great athletes won a large number of the (Gold) medals of a certain sport or event, and sometimes his/her country's medal tally greatly depend on him/her during his/her athlete career. We will also explain how this can inform country Olympic committees.

2 Assumptions and Notations

2.1 Assumptions and Justifications

To simplify problem, make it convenient to construct simulation model and ensure it's effectiveness, following basic assumptions are made, each of which is properly justified.

- **Data Preprocess and Country Mapping**.

  Data is cleaned and preprocessed to make it more suitable. We set the ice sports aside, combine various teams of one country, and map the countries which no longer exist to the current existing country. It's a rather difficult task since countries have changed a lot during the past century, but it's necessary to stick to the current world map.

- **Relationship between Medal counts and Historical data**.

  Most data provided is examined to be accurate and reliable, although some may not be the same as the data provided from other sources. Few of the data is examined and changed since it's too faraway from reality. Medal counts are also assumed to be related to multiple variables:years, gender, athletes and number of sports, diciplines, events,etc.

- **Countries that will possibly take part in the 2028 Olympics**.

  Countries that have taken part in 2024 Olympics will also take part in 2028 Olympics. There's no reason for them not to attend the Olympics, and it's impossible to estimate medal of new countries if they've never taken part in the Olympics using historical data. Russia is also assumed to participate in the 2028 Olympics, whether the athletes will be able to compete under the Russian flag or not, their country is called Russia.

- **Possible Sports in 2028**.

  There's said to be five new sports in the 2028 Olympics, but we set them aside because we don't have enough data to predict the medal counts of these events. Similarly, we set aside the sports that will be removed from the Olympics in 2028, and assume that most sports are like that in 2024 or their most common appearance in history(Such as gymnastics).

Other details will be stated in the following sections when we discuss the specific models.

2.2 Notations

Symbols	Definitions	Symbols	Definitions
h	aaaaaaaaaaaaaaaaaaa	a	aaaaaaaaaaaaaaaaaaaaa
k			
c_p			
ρ			

Define the Main parameters. Other Specific notations will be listed and explained later.

3 Data Preprocessing

3.1 Data Cleaning

- **Garbled Code**

Figure 1: bad data

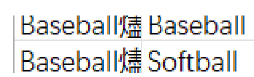

Figure 2: bad text

We carefully examine the data and find that there are some garbled codes in the data, which may be caused by the encoding of the data. They are replaced by the correct text.

- **Null, Duplicated Values and Renaming**

Listing 1: Data Cleaning

```
data.fillna(0, inplace=True)
data.drop_duplicates(inplace=True)
athletes.rename(columns={'Team': 'Country'}, inplace=True)
```

Team are renamed to Country in the athletes data so that there won't be Germany-1.

- **Data Splitting**

Listing 2: Data Splitting

```
    data.fillna(0, inplace=True)
    data.drop_duplicates(inplace=True)
    athletes.rename(columns={'Team': 'Country'}, inplace=True)
```

- **Remove unnecessary data**

Listing 3: Remove Ice Sports

```
    ice_sports = ['Figure Skating', 'Ice Hockey']
    programs = programs[~programs['Sport'].isin(ice_sports)]
    athletes = athletes[~athletes['Sport'].isin(ice_sports)]
```

Listing 4: Remove Program from 1906

```
# Remove program from the year 1906
program = program[program['Year'] != 1906]
```

3.2 Country Mapping

Listing 5: Country Mapping

```
country_mapping = {# map due to the change of country in history
    'Soviet Union': 'Russia','Unified Team': 'Russia',
    'West Germany': 'Germany','East Germany': 'Germany',
    'Yugoslavia': 'Serbia',
    'Bohemia': 'Czech Republic','Czechoslovakia': 'Czech Republic',
    'Virgin Islands': 'United States',
}
# NOC mapping is similar.Only part of the country_mapping is shown here.There're
    ↪ also map between NOC and country.
athletes['NOC'] = athletes['NOC'].replace(noc_mapping)
medals['NOC'] = medals['NOC'].replace(country_mapping)
```

3.3 Primary processed data: The 'Top 10s'

Top 10 Countries by Medal Count

NOC	Gold	Silver	Bronze	Total
United States	1105	880	780	2765
Russia	588	487	478	1553
Germany	457	475	503	1435
Great Britain	299	339	343	981
France	240	278	299	817
China	303	226	198	727
Italy	230	205	228	663
Australia	182	192	226	600
Japan	189	162	193	544
Hungary	187	162	182	531

Figure 3: top 10 countries

Top 10 Athletes by Medal Count

Name	Gold	Silver	Bronze	Total
Michael Ii	23	3	2	28
Larysa (diriy-)	9	5	4	18
Nikolay Andrianov	7	5	3	15
Charles Jr.	9	4	2	15
John Jr.	9	1	3	13
Borys Shakhlin	7	4	2	13
Edoardo Mangiarotti	6	5	2	13
Takashi Ono	5	4	4	13
Sawao Kato	8	3	1	12
Dara -minas)	4	4	4	12

Figure 4: top 10 athletes

4 Model Overview

In order to solve those problems, we will proceed as follows:

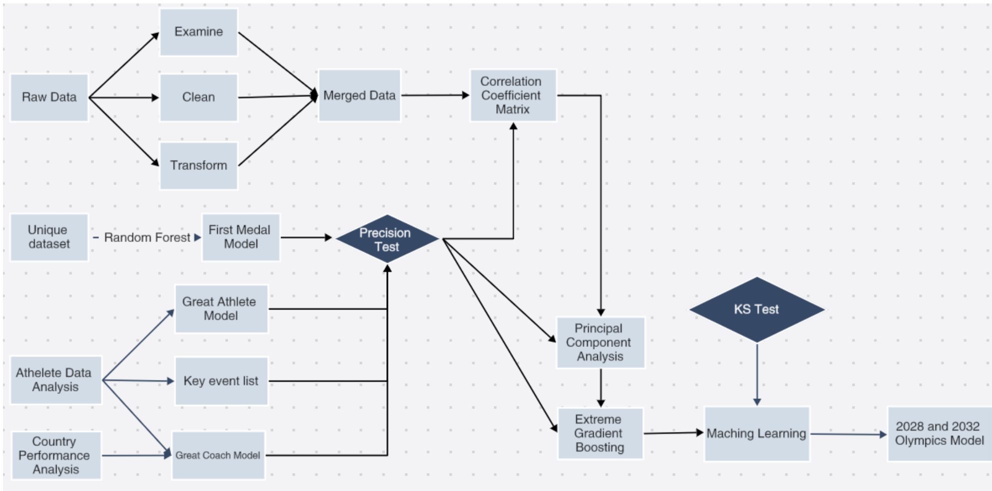

Figure 5: Flow chart of our work

- **Presenting our model**. In order to investigate the problem deeper, we divide our task into four models: **Predicting the Medal of the 2028 LA Olympics**, **Predicting First Medal Countries**, **Great Coach Effect**, **Key Events and Great Athletes**.
- **Data Processing**.First, We examine, clean and transform the data given and merge them. Then the data is put into the Correlation Coefficient Matrix for further machine learning.
- **First Medal Model**.Next, We use **Random Forest** to train the model, giving prediciton of first medal countries.
- **Performance Analysis**.Then, we calculate the performance of Athletes and countries (on certain sports/events), forming the **Key Event Model** and the **Great Coach Model**.
- **Machine Learning**. Finally, we use **KS Test** to ensure the precision of the final model.

5 Task I : Predicting the Medal of the 2028 LA Olympics

5.1 Model Establishment

Since we want to predict future medal list on the basis of past data, we merge the given data and do some preprocessing. We combine the data with the same **Year** and **NOC** together.

We use **Correlation Coefficient Matrix** and **Principal Component Analysis**, split the data into training and testing parts, and use **Extreme Gradient Boosting methods** to train features and target matrix.

We also use **KS test** to ensure the effectiveness of the model.After the model is established, we use it to predict the 2028 and 2032 Olympics.

What's important is that we add the factor of whether being a host city/country or not in to the training model so that the final result can take this factor into consideration. For example, in the coming LA Olympics, the USA is likely to gain more medals due to the **"hosting effect"**.

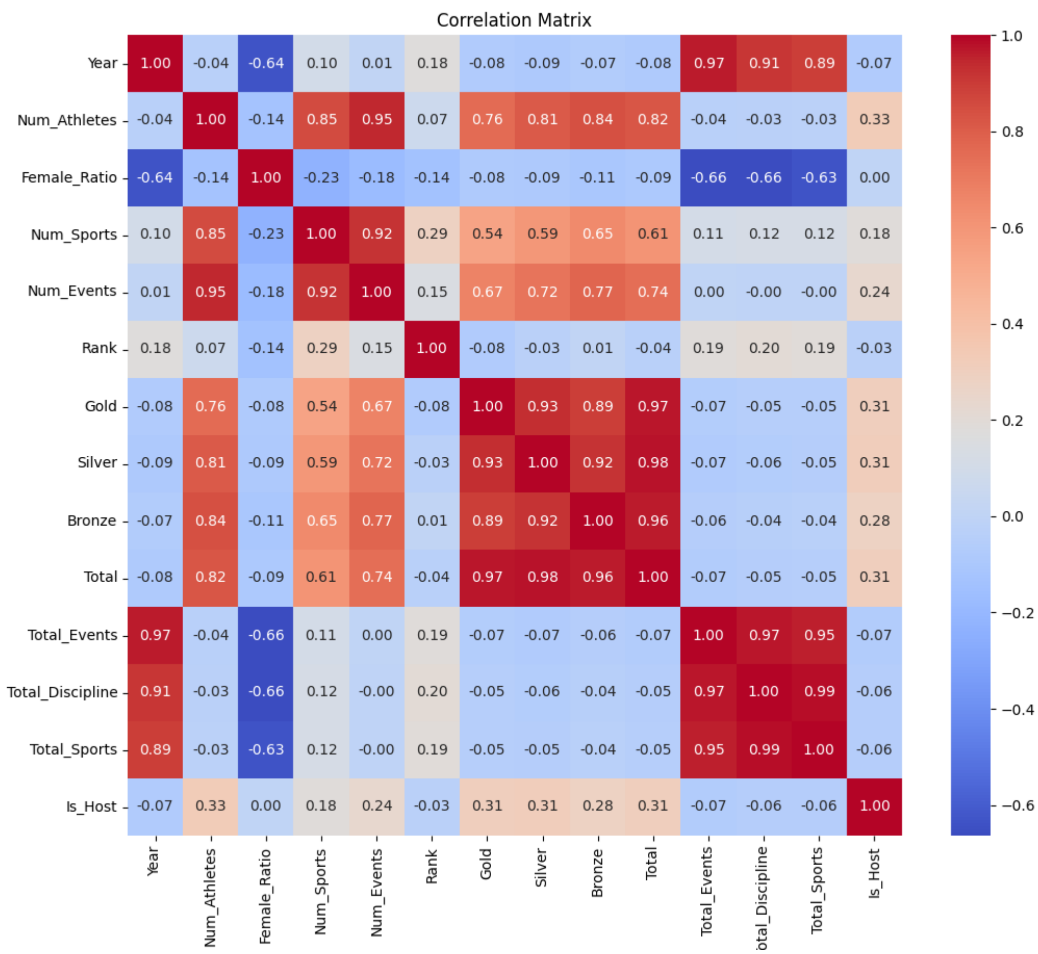

Figure 6: Correlation Matrix

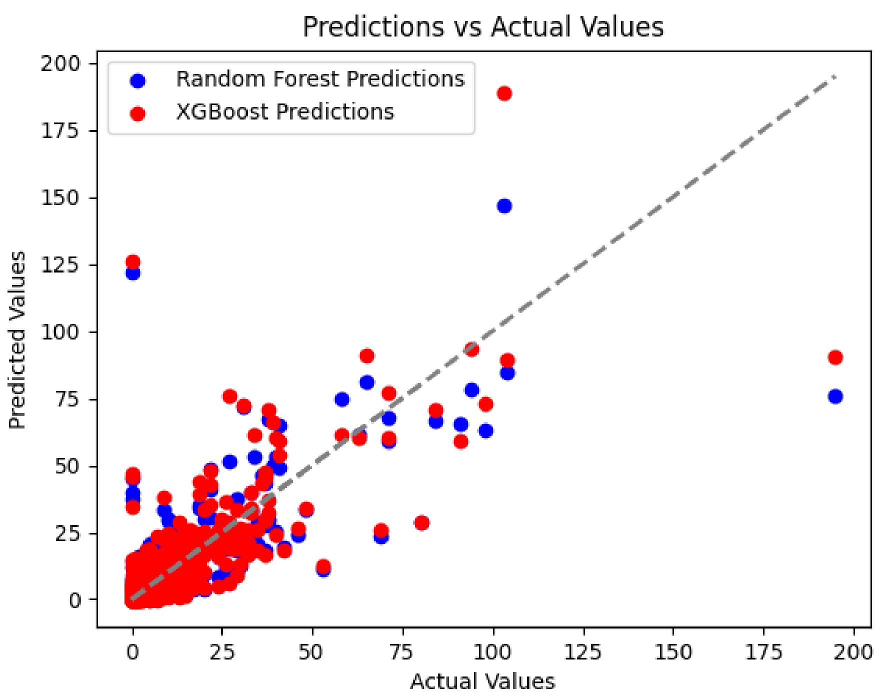

Figure 7: Precision of the estimation

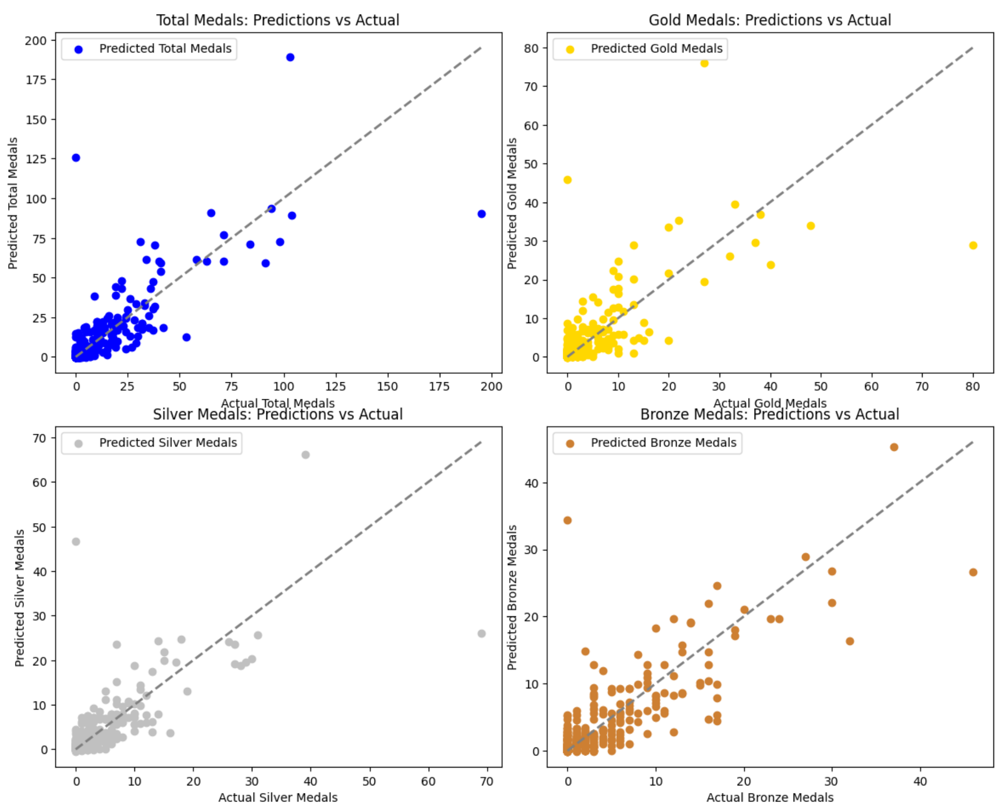

Figure 8: Precision of the estimation(Gold, Silver, Bronze and Total)

```
Random Forest Predictions: [[6.00000000e-01 2.50000000e-01 1.90000000e-01 1.60000000e-01]
 [1.02000000e+00 1.70000000e-01 3.50000000e-01 5.00000000e-01]
 [6.19000000e+00 1.28000000e+00 2.64000000e+00 2.27000000e+00]
 ...
 [1.05000000e+00 7.00000000e-02 3.90000000e-01 5.90000000e-01]
 [4.85714286e-03 4.85714286e-03 0.00000000e+00 0.00000000e+00]
 [2.99000000e+00 7.30000000e-01 1.27000000e+00 9.90000000e-01]]
XGBoost Predictions: [[ 5.1203884e-02  3.2490236e-03  1.3189411e-04  5.9093786e-03]
 [-2.0074600e-02  1.5166138e-02  2.2303849e-03  3.3099253e-02]
 [ 6.2406716e+00  1.2808298e+00  3.1602407e+00  2.2505367e+00]
 ...
 [ 5.8379167e-01 -5.4412475e-03  2.2964623e-02  3.4039971e-01]
 [ 2.9888877e-04  1.2590903e-03 -3.9039849e-05 -5.8104652e-05]
 [ 3.4480846e+00  9.0951747e-01  1.3775554e+00  1.0207008e+00]]
K-S Test Statistic: [0.3       0.4953125 0.390625  0.4265625], P-Value: [8.25480975e-26 1.16773287e-71 5.95698276e-44 1.29633612e-52]
Random Forest MSE: 27.695270017868456, R^2: 0.6306700795983087
XGBoost MSE: 30.791641235351562, R^2: 0.5817188024520874
XGBoost Total Medals - MSE: 86.20039085781896, R^2: 0.6263195818297187
XGBoost Gold Medals - MSE: 18.183467512401762, R^2: 0.45440816844255594
XGBoost Silver Medals - MSE: 11.543446173187839, R^2: 0.5503359404587108
XGBoost Bronze Medals - MSE: 7.239262598410515, R^2: 0.6958110484049175
```

Figure 9: Specific data during training and estimation

According to the results graphs following, we can see that our model is of good accuracy, and have a good performance on test set.

From Figure2, we can see that the prediction of XGBoost outperformed the prediciton of Random Forest Predicitons, which is one of the reason for our choice of model.

As to the specific principle of XGBoost:

$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

The loss function and the objective function are as follows:

$L(\hat{y}_i^{(t-1)}, y_i)$ loss function y_i prediction from the first to the t-1th tree $\hat{y}_i^{(t-1)}$

The complete objective function (loss function plus regularization term) can be represented as:

$\mathcal{L}^{(t)} = \sum_{i=1}^{n} L(\hat{y}_i^{(t-1)}, y_i) + \Omega(f_t)$

where $\Omega(f_t)$ is the model complexity of the t-th tree, which can be expressed as:

$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$

where T is the number of nodes in the tree, w_j is the weight of each node, and γ and λ are regularization parameters.

6 Task II : Predicting First Medal Countries

6.1 Problem Analysis

More than 60 countries have still yet to win an Olympic medal.

6.2 Model Establishment and Solution

6.3 Result Analysis

7 Task III : Great Coach Effect

8 Task IV : Key Events and Great Athletes

8.1 Key Events

8.1.1 Calculation Methods

To identify key events, we will use the proportion of medals from these events among the whole medal tally as a main factor.

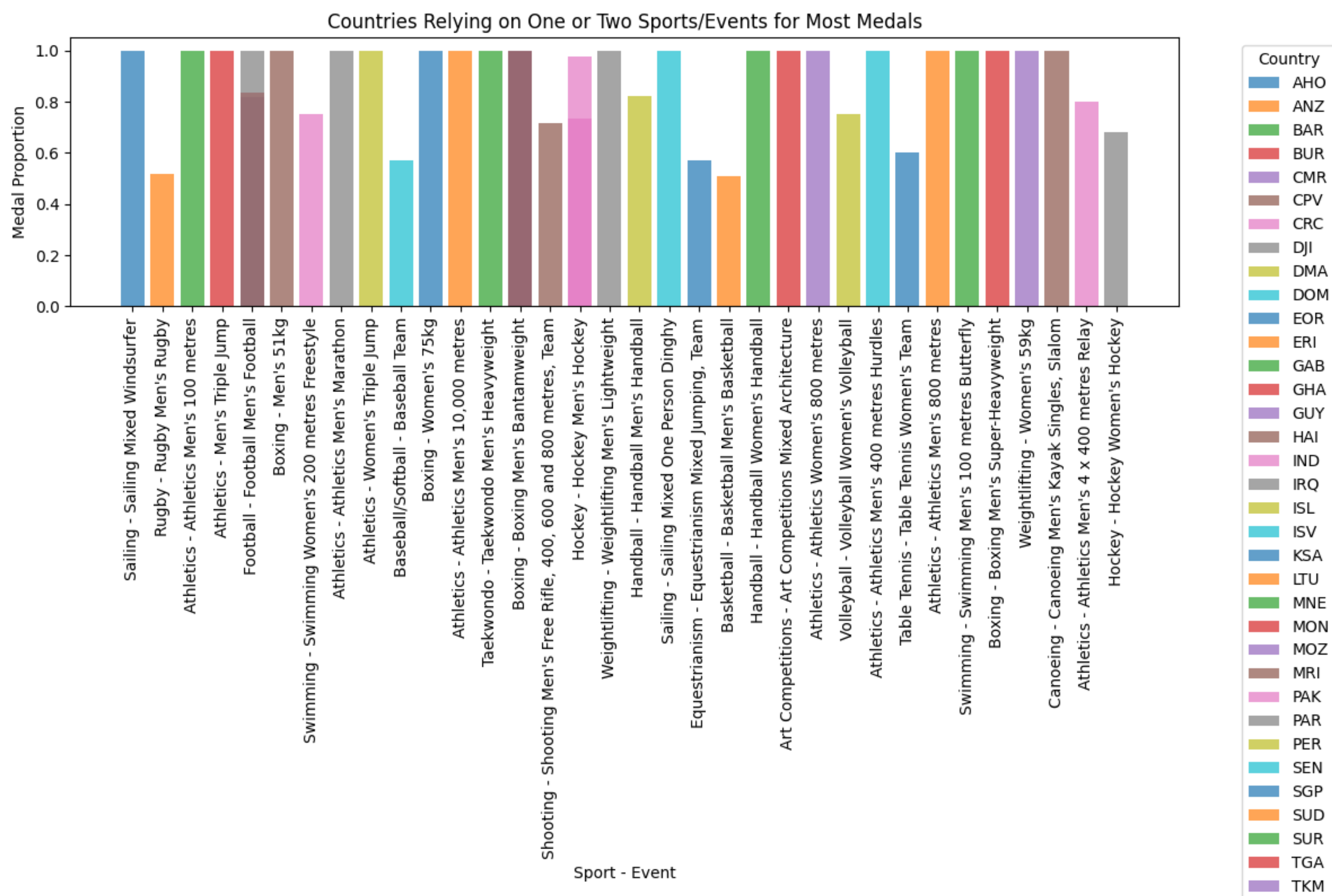

Figure 10: Countries with biased medal distribution

From this graph, we can see that many countries rely on one or two sports/events for their medal tally. This can be inspiring for the countries' Olympics committees.

8.2 Great Athletes

8.2.1 Calculation Methods

To identify Great Athletes, we would like to see those who secured almost all the (Gold) medals in their careers. Often, these Athletes play a significant role in their countries' Olympics team for winning medals, or at least in this certain sport/event.

From the graph we can see that **Great Athletes** can impact the event and their countries performance greatly, which can be also inspiring for the countries' Olympics committees, for instance, how to cope with the possible downfall of the national team's performance after these great stars retired.

9 Other Insights

9.1 Try to predict the medal of the 2032 Olympics

9.2 Greatness and the strategy of the countries' Olympic committees

10 Strength and Weakness

10.1 Strength

- We analyze the problem based on thermodynamic formulas and laws, so that the model we established is of great validity.
- Our model is fairly robust due to our careful corrections in consideration of real-life situations and detailed sensitivity analysis.
- Via Fluent software, we simulate the time field of different areas throughout the bathtub. The outcome is vivid for us to understand the changing process.
- We come up with various criteria to compare different situations, like water consumption and the time of adding hot water. Hence an overall comparison can be made according to these criteria.
- Besides common factors, we still consider other factors, such as evaporation and radiation heat transfer. The evaporation turns out to be the main reason of heat loss, which corresponds with other scientist's experimental outcome.

10.2 Weakness

- Having knowing the range of some parameters from others' essays, we choose a value from them to apply in our model. Those values may not be reasonable in reality.
- Although we investigate a lot in the influence of personal motions, they are so complicated that need to be studied further.
- Limited to time, we do not conduct sensitivity analysis for the influence of personal surface area.

10.3 Further Discussions

References

[1] Nielsen. *2024 virtual-medal-table-forecast*. `https://www.nielsen.com/news-center/2024/virtual-medal-table-forecast/`. 2024.

Appendices

Appendix A Code for Predicting 2028 Olympics Medal

Input Python source:

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from scipy.stats import ks_2samp
import numpy as np
import matplotlib.pyplot as plt
import chardet
from sklearn.metrics import mean_squared_error, r2_score
import seaborn as sns
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize

#
file_dict_path = r".\2025_Problem_C_Data\data_dictionary.csv"
athletes_file_path = r".\2025_Problem_C_Data\summerOly_athletes.csv"
hosts_file_path = r".\2025_Problem_C_Data\summerOly_hosts.csv"
medals_file_path = r".\2025_Problem_C_Data\summerOly_medal_counts.csv"
programs_file_path = r".\2025_Problem_C_Data\summerOly_programs.csv"

#
def detect_encoding(file_path):
    with open(file_path, 'rb') as f:
        raw_data = f.read()
        result = chardet.detect(raw_data)
        return result['encoding']

#   Pandas  CSV
def read_csv(file_path):
    encoding = detect_encoding(file_path)
    return pd.read_csv(file_path, encoding=encoding)

#
data_dict = read_csv(file_dict_path)
athletes = read_csv(athletes_file_path)
hosts = read_csv(hosts_file_path)
medal_counts = read_csv(medals_file_path)
programs = read_csv(programs_file_path)
```

```
country_mapping = {
    'Soviet Union': 'Russia',
    'West Germany': 'Germany',
    'East Germany': 'Germany',
    'Yugoslavia': 'Serbia',
    'Czechoslovakia': 'Czech Republic',
    'Bohemia': 'Czech Republic',
    'Russian Empire': 'Russia',
    'United Team of Germany': 'Germany',
    'Unified Team': 'Russia',
    'Serbia and Montenegro': 'Serbia',
    'Netherlands Antilles': 'Netherlands',
    'Virgin Islands': 'United States',
    'West Indies Federation': 'United States',
    'ROC': 'Russia',
    'LIB': 'Liberia',
    'Libya': 'Liberia',
}

noc_mapping = {
    'URS': 'RUS',
    'EUA': 'GER',
    'FRG': 'GER',
    'GDR': 'GER',
    'YUG': 'SRB',
    'TCH': 'CZE',
    'BOH': 'CZE',
    'EUN': 'RUS',
    'SCG': 'SRB',
    'ANZ': 'AUS',
    'NBO': 'KEN',
    'WIF': 'USA',
    'IOP': 'IOA',
    'ROC': 'RUS',
    'EOR': 'RUS',
}

country_codes = {
    'EOR': 'Russia',
    'ROC': 'Russia',
    'AFG': 'Afghanistan',
    'ALB': 'Albania',
    'ALG': 'Algeria',
    'AND': 'Andorra',
    'ANG': 'Angola',
    'ANT': 'Antigua and Barbuda',
```

```
'ARG': 'Argentina',
'ARM': 'Armenia',
'ARU': 'Aruba',
'ASA': 'American Samoa',
'AUS': 'Australia',
'AUT': 'Austria',
'AZE': 'Azerbaijan',
'BAH': 'Bahamas',
'BAN': 'Bangladesh',
'BAR': 'Barbados',
'BDI': 'Burundi',
'BEL': 'Belgium',
'BEN': 'Benin',
'BER': 'Bermuda',
'BHU': 'Bhutan',
'BIH': 'Bosnia and Herzegovina',
'BIZ': 'Belize',
'BLR': 'Belarus',
'BOL': 'Bolivia',
'BOT': 'Botswana',
'BRA': 'Brazil',
'BRN': 'Bahrain',
'BRU': 'Brunei',
'BUL': 'Bulgaria',
'BUR': 'Burkina Faso',
'CAF': 'Central African Republic',
'CAM': 'Cambodia',
'CAN': 'Canada',
'CAY': 'Cayman Islands',
'CGO': 'Congo',
'CHA': 'Chad',
'CHI': 'Chile',
'CHN': 'China',
'CIV': 'Ivory Coast',
'CMR': 'Cameroon',
'COD': 'Congo',
'COK': 'Cook Islands',
'COL': 'Colombia',
'COM': 'Comoros',
'CPV': 'Cape Verde',
'CRC': 'Costa Rica',
'CRO': 'Croatia',
'CUB': 'Cuba',
'CYP': 'Cyprus',
'CZE': 'Czech Republic',
'DEN': 'Denmark',
'DJI': 'Djibouti',
```

```
    'DMA': 'Dominica',
    'DOM': 'Dominican Republic',
    'ECU': 'Ecuador',
    'EGY': 'Egypt',
    'ERI': 'Eritrea',
    'ESA': 'El Salvador',
    'ESP': 'Spain',
    'EST': 'Estonia',
    'ETH': 'Ethiopia',
    'FIJ': 'Fiji',
    'FIN': 'Finland',
    'FRA': 'France',
    'FSM': 'Micronesia',
    'GAB': 'Gabon',
    'GAM': 'Gambia',
    'GBR': 'Great Britain',
    'GBS': 'Guinea-Bissau',
    'GEO': 'Georgia',
    'GEQ': 'Equatorial Guinea',
    'GER': 'Germany',
    'GHA': 'Ghana',
    'GRE': 'Greece',
    'GRN': 'Grenada',
    'GUA': 'Guatemala',
    'GUI': 'Guinea',
    'GUM': 'Guam',
    'GUY': 'Guyana',
    'HAI': 'Haiti',
    'HKG': 'Hong Kong',
    'HON': 'Honduras',
    'HUN': 'Hungary',
    'INA': 'Indonesia',
    'IND': 'India',
    'IRI': 'Iran',
    'IRL': 'Ireland',
    'IRQ': 'Iraq',
    'ISL': 'Iceland',
    'ISR': 'Israel',
    'ISV': 'Virgin Islands',
    'ITA': 'Italy',
    'IVB': 'British Virgin Islands',
    'JAM': 'Jamaica',
    'JOR': 'Jordan',
    'JPN': 'Japan',
    'KAZ': 'Kazakhstan',
    'KEN': 'Kenya',
    'KGZ': 'Kyrgyzstan',
```

```
    'KIR': 'Kiribati',
    'KOR': 'South Korea',
    'KOS': 'Kosovo',
    'KSA': 'Saudi Arabia',
    'KUW': 'Kuwait',
    'LAO': 'Laos',
    'LAT': 'Latvia',
    'LBA': 'Lebanon',
    'LBR': 'Liberia',
    'LCA': 'Saint Lucia',
    'LES': 'Lesotho',
    'LIE': 'Liechtenstein',
    'LTU': 'Lithuania',
    'LUX': 'Luxembourg',
    'LIB': 'Liberia',
    'MAD': 'Madagascar',
    'MAR': 'Morocco',
    'MAS': 'Malaysia',
    'MAW': 'Malawi',
    'MDA': 'Moldova',
    'MDV': 'Maldives',
    'MEX': 'Mexico',
    'MGL': 'Mongolia',
    'MHL': 'Marshall Islands',
    'MKD': 'North Macedonia',
    'MLI': 'Mali',
    'MLT': 'Malta',
    'MNE': 'Montenegro',
    'MON': 'Monaco',
    'MOZ': 'Mozambique',
    'MRI': 'Mauritius',
    'MTN': 'Mauritania',
    'MYA': 'Myanmar',
    'NAM': 'Namibia',
    'NCA': 'Nicaragua',
    'NED': 'Netherlands',
    'NEP': 'Nepal',
    'NGR': 'Nigeria',
    'NIG': 'Niger',
    'NOR': 'Norway',
    'NRU': 'Nauru',
    'NZL': 'New Zealand',
    'OMA': 'Oman',
    'PAK': 'Pakistan',
    'PAN': 'Panama',
    'PAR': 'Paraguay',
    'PER': 'Peru',
```

```
    'PHI': 'Philippines',
    'PLE': 'Palestine',
    'PLW': 'Palau',
    'PNG': 'Papua New Guinea',
    'POL': 'Poland',
    'POR': 'Portugal',
    'PRK': 'North Korea',
    'PUR': 'Puerto Rico',
    'QAT': 'Qatar',
    'ROU': 'Romania',
    'RSA': 'South Africa',
    'RUS': 'Russia',
    'RWA': 'Rwanda',
    'SAM': 'Samoa',
    'SEN': 'Senegal',
    'SEY': 'Seychelles',
    'SGP': 'Singapore',
    'SIN': 'Singapore',
    'SKN': 'Saint Kitts and Nevis',
    'SLE': 'Sierra Leone',
    'SLO': 'Slovenia',
    'SMR': 'San Marino',
    'SOL': 'Solomon Islands',
    'SRB': 'Serbia',
    'SRI': 'Sri Lanka',
    'STP': 'Sao Tome and Principe',
    'SUD': 'Sudan',
    'SUI': 'Switzerland',
    'SUR': 'Suriname',
    'SVK': 'Slovakia',
    'SWE': 'Sweden',
    'SWZ': 'Eswatini',
    'SYR': 'Syria',
    'TAN': 'Tanzania',
    'TGA': 'Tonga',
    'THA': 'Thailand',
    'TJK': 'Tajikistan',
    'TKM': 'Turkmenistan',
    'TLS': 'Timor-Leste',
    'TOG': 'Togo',
    'TPE': 'Chinese Taipei',
    'TTO': 'Trinidad and Tobago',
    'TUN': 'Tunisia',
    'TUR': 'Turkey',
    'TUV': 'Tuvalu',
    'UAE': 'United Arab Emirates',
    'UGA': 'Uganda',
```

```
    'UKR': 'Ukraine',
    'URU': 'Uruguay',
    'USA': 'United States',
    'UZB': 'Uzbekistan',
    'VAN': 'Vanuatu',
    'VEN': 'Venezuela',
    'VIE': 'Vietnam',
    'VIN': 'Saint Vincent and the Grenadines',
    'YEM': 'Yemen',
    'ZAM': 'Zambia',
    'ZIM': 'Zimbabwe'
}



athletes['NOC'] = athletes['NOC'].replace(noc_mapping)
medal_counts['NOC'] = medal_counts['NOC'].replace(country_mapping)

# Split Host in hosts.csv into City and Country
hosts[['City', 'NOC']] = hosts['Host'].str.split(',', expand=True)
hosts['NOC'] = hosts['NOC'].str.strip()

# Map NOC in athletes.csv to countries
athletes['NOC'] = athletes['NOC'].map(country_codes).fillna(athletes['NOC'])

# Preprocess athletes data
athletes['Sex'] = athletes['Sex'].map({'M': 1, 'F': 0})
athletes_agg = athletes.groupby(['Year', 'NOC']).agg({
    'Name': lambda x: x.nunique(),
    'Sex': lambda x: x.mean(),
    'Sport': lambda x: x.nunique(),
    'Event': lambda x: x.nunique()
}).reset_index()
athletes_agg.rename(columns={'Name': 'Num_Athletes', 'Sex': 'Female_Ratio',
    ↪ 'Sport': 'Num_Sports', 'Event': 'Num_Events'}, inplace=True)



athletes_agg.to_csv('./2025_Problem_C_Data/athletes_agg.csv')

# Convert 'Year' column to int in medal_counts
medal_counts['Year'] = medal_counts['Year'].astype(int)

# Merge athletes_agg and medal_counts
data = pd.merge(athletes_agg, medal_counts, on=['Year', 'NOC'], how='left')
```

```
# Read specific rows and columns from programs.csv
programs_sum = pd.read_csv(programs_file_path, skiprows=lambda x: x not in [0,
    ↪ 72, 73, 74], usecols=range(4, programs.shape[1]))

# Transform the data into the required format
programs_sum = programs_sum.transpose().reset_index()
programs_sum.columns = ['Year', 'Total_Events', 'Total_Discipline',
    ↪ 'Total_Sports']
# Convert 'Year' column to int in programs_sum
programs_sum['Year'] = programs_sum['Year'].astype(int)

# Merge programs_sum with data on Year
data = pd.merge(data, programs_sum, on='Year', how='left')

# Determine if the country is the host for each year
data['Is_Host'] = data.apply(lambda row: 1 if row['NOC'] in hosts[hosts['Year']
    ↪ == row['Year']]['NOC'].values else 0, axis=1)

data = data.fillna(0)

data.to_csv('./2025_Problem_C_Data/data.csv')




# Prepare features and target with additional variables
X = data[['Year', 'Is_Host', 'Num_Athletes', 'Female_Ratio', 'Num_Sports',
    ↪ 'Num_Events','Total_Events','Total_Discipline','Total_Sports']]
y = data[['Total','Gold','Silver','Bronze']].apply(pd.to_numeric,
    ↪ errors='coerce')


# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    ↪ random_state=42)

# Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)

# Create new variable matrix for XGBoost
X_train_rf = X_train.copy()
X_test_rf = X_test.copy()
for i, col in enumerate(['Total', 'Gold', 'Silver', 'Bronze']):
    X_train_rf[f'RF_Predictions_{col}'] = rf_model.predict(X_train)[:, i]
    X_test_rf[f'RF_Predictions_{col}'] = rf_predictions[:, i]
```

```
# XGBoost model
xgb_model = XGBRegressor(n_estimators=100, random_state=42)
xgb_model.fit(X_train_rf, y_train)
xgb_predictions = xgb_model.predict(X_test_rf)

# K-S Test
ks_stat, p_value = ks_2samp(y_test, xgb_predictions)

# Print results
print(f"Random Forest Predictions: {rf_predictions}")
print(f"XGBoost Predictions: {xgb_predictions}")
print(f"K-S Test Statistic: {ks_stat}, P-Value: {p_value}")

# # Predict 2028 medals, golds, silver, bronze
# future_data = pd.DataFrame({
#     'Year': [2028],
#     'Is_Host': [1],
#     'Num_Athletes': [613], # Placeholder value
#     'Female_Ratio': [0.463], # Placeholder value
#     'Num_Sports': [47], # Placeholder value
#     'Num_Events': [234],  # Placeholder value
#     'Total_Events': [329], # Placeholder value
#     'Total_Discipline': [48], # Placeholder value
#     'Total_Sports': [32] # Placeholder value
# })
# rf_predictions_future = rf_model.predict(future_data).reshape(1, -1)
# future_data['RF_Predictions_Total'] = rf_predictions_future[:, 0]
# future_data['RF_Predictions_Gold'] = rf_predictions_future[:, 1]
# future_data['RF_Predictions_Silver'] = rf_predictions_future[:, 2]
# future_data['RF_Predictions_Bronze'] = rf_predictions_future[:, 3]

# # Ensure future_data has the same columns as X_train_rf
# future_data_rf = future_data[['Year', 'Is_Host', 'Num_Athletes',
   ↪ 'Female_Ratio', 'Num_Sports', 'Num_Events', 'Total_Events',
   ↪ 'Total_Discipline', 'Total_Sports',
#                           'RF_Predictions_Total', 'RF_Predictions_Gold',
   ↪ 'RF_Predictions_Silver', 'RF_Predictions_Bronze']]

# future_predictions = xgb_model.predict(future_data_rf)
# print(f"Predicted Medals for 2028: Total: {future_predictions[0][0]}, Gold:
   ↪ {future_predictions[0][1]}, Silver: {future_predictions[0][2]}, Bronze:
   ↪ {future_predictions[0][3]}")


# Predict medals for 2028 for all countries
future_data_all = data[(data['Year'] == 2024) & (data['NOC'] != 'France') |
```

```
    ↪ ((data['Year'] == 2020) & (data['NOC'] == 'France')) | ((data['Year'] ==
    ↪ 2016) & (data['NOC'] == 'Russia'))].copy()

future_data_all['Year'] = 2028


# Predict using Random Forest
rf_predictions_all = rf_model.predict(future_data_all[['Year', 'Is_Host',
    ↪ 'Num_Athletes', 'Female_Ratio', 'Num_Sports', 'Num_Events',
    ↪ 'Total_Events', 'Total_Discipline', 'Total_Sports']])
future_data_all['RF_Predictions_Total'] = rf_predictions_all[:, 0]
future_data_all['RF_Predictions_Gold'] = rf_predictions_all[:, 1]
future_data_all['RF_Predictions_Silver'] = rf_predictions_all[:, 2]
future_data_all['RF_Predictions_Bronze'] = rf_predictions_all[:, 3]

# Ensure future_data_all has the same columns as X_train_rf
future_data_rf_all = future_data_all[['Year', 'Is_Host', 'Num_Athletes',
    ↪ 'Female_Ratio', 'Num_Sports', 'Num_Events', 'Total_Events',
    ↪ 'Total_Discipline', 'Total_Sports',
                                  'RF_Predictions_Total', 'RF_Predictions_Gold',
                                      ↪ 'RF_Predictions_Silver',
                                      ↪ 'RF_Predictions_Bronze']]

# Predict using XGBoost
future_predictions_all = xgb_model.predict(future_data_rf_all)

# Aggregate predicted medal counts for each country
future_data_all['Predicted_Gold'] = future_predictions_all[:, 1]
future_data_all['Predicted_Silver'] = future_predictions_all[:, 2]
future_data_all['Predicted_Bronze'] = future_predictions_all[:, 3]

# Aggregate medal counts for each country
future_medal_totals = future_data_all.groupby('NOC')[['Predicted_Gold',
    ↪ 'Predicted_Silver', 'Predicted_Bronze']].sum().reset_index()

# Sort by predicted total medals and save to CSV
future_medal_totals['Predicted_Total'] = future_medal_totals['Predicted_Gold'] +
    ↪ future_medal_totals['Predicted_Silver'] +
    ↪ future_medal_totals['Predicted_Bronze']
future_medal_totals_sorted =
    ↪ future_medal_totals.sort_values(by='Predicted_Total', ascending=False)
future_medal_totals_sorted.to_csv('./2025_Problem_C_Data/future_medal_totals_2028.csv',
    ↪ index=False)

# Sort by predicted total medals and get top 10 countries
top_10_future_countries = future_medal_totals_sorted.head(10)
```

```
# Plot the predicted medal counts for the top 10 countries in 2028 with
    ↪ horizontal stacked bars
fig, ax = plt.subplots(figsize=(12, 8))

# Plot stacked bars for each medal type
ax.barh(top_10_future_countries['NOC'],
    ↪ top_10_future_countries['Predicted_Gold'], color='gold', label='Gold')
ax.barh(top_10_future_countries['NOC'],
    ↪ top_10_future_countries['Predicted_Silver'],
    ↪ left=top_10_future_countries['Predicted_Gold'], color='silver',
    ↪ label='Silver')
ax.barh(top_10_future_countries['NOC'],
    ↪ top_10_future_countries['Predicted_Bronze'],
    ↪ left=top_10_future_countries['Predicted_Gold'] +
    ↪ top_10_future_countries['Predicted_Silver'], color='#cd7f32',
    ↪ label='Bronze')

# Add data labels
for i in range(len(top_10_future_countries)):
    ax.text(top_10_future_countries['Predicted_Gold'].iloc[i] / 2, i,
        ↪ int(top_10_future_countries['Predicted_Gold'].iloc[i]), va='center',
        ↪ ha='center', color='black')
    ax.text(top_10_future_countries['Predicted_Gold'].iloc[i] +
        ↪ top_10_future_countries['Predicted_Silver'].iloc[i] / 2, i,
        ↪ int(top_10_future_countries['Predicted_Silver'].iloc[i]), va='center',
        ↪ ha='center', color='black')
    ax.text(top_10_future_countries['Predicted_Gold'].iloc[i] +
        ↪ top_10_future_countries['Predicted_Silver'].iloc[i] +
        ↪ top_10_future_countries['Predicted_Bronze'].iloc[i] / 2, i,
        ↪ int(top_10_future_countries['Predicted_Bronze'].iloc[i]), va='center',
        ↪ ha='center', color='black')

plt.xlabel('Number of Medals')
plt.ylabel('Country')
plt.title('Top 10 Countries by Predicted Medals in 2028')
plt.legend(loc='upper right')
plt.gca().invert_yaxis()
plt.show()


# Calculate Mean Squared Error and R^2 for Random Forest
mse_rf = mean_squared_error(y_test, rf_predictions)
r2_rf = r2_score(y_test, rf_predictions)

# Calculate Mean Squared Error and R^2 for XGBoost
mse_xgb = mean_squared_error(y_test, xgb_predictions)
r2_xgb = r2_score(y_test, xgb_predictions)
```

```
# Print regression metrics
print(f"Random Forest MSE: {mse_rf}, R^2: {r2_rf}")
print(f"XGBoost MSE: {mse_xgb}, R^2: {r2_xgb}")

# Plot predictions vs actual values
plt.figure()
plt.scatter(y_test, rf_predictions, color='blue', label='Random Forest
    ↪ Predictions')
plt.scatter(y_test, xgb_predictions, color='red', label='XGBoost Predictions')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
    ↪ color='gray', lw=2, linestyle='--')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Predictions vs Actual Values')
plt.legend(loc='upper left')
plt.show()

# Plot correlation matrix without 'NOC' column
plt.figure(figsize=(12, 10))
correlation_matrix = data.drop(columns=['NOC']).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Calculate Mean Squared Error and R^2 for each target variable
mse_total = mean_squared_error(y_test['Total'], xgb_predictions[:, 0])
r2_total = r2_score(y_test['Total'], xgb_predictions[:, 0])

mse_gold = mean_squared_error(y_test['Gold'], xgb_predictions[:, 1])
r2_gold = r2_score(y_test['Gold'], xgb_predictions[:, 1])

mse_silver = mean_squared_error(y_test['Silver'], xgb_predictions[:, 2])
r2_silver = r2_score(y_test['Silver'], xgb_predictions[:, 2])

mse_bronze = mean_squared_error(y_test['Bronze'], xgb_predictions[:, 3])
r2_bronze = r2_score(y_test['Bronze'], xgb_predictions[:, 3])

# Print regression metrics for each target variable
print(f"XGBoost Total Medals - MSE: {mse_total}, R^2: {r2_total}")
print(f"XGBoost Gold Medals - MSE: {mse_gold}, R^2: {r2_gold}")
print(f"XGBoost Silver Medals - MSE: {mse_silver}, R^2: {r2_silver}")
print(f"XGBoost Bronze Medals - MSE: {mse_bronze}, R^2: {r2_bronze}")
# Plot predictions vs actual values for each target variable
fig, axs = plt.subplots(2, 2, figsize=(14, 12))

# Total Medals
```

```
axs[0, 0].scatter(y_test['Total'], xgb_predictions[:, 0], color='blue',
    ↪ label='Predicted Total Medals')
axs[0, 0].plot([y_test['Total'].min(), y_test['Total'].max()],
    ↪ [y_test['Total'].min(), y_test['Total'].max()], color='gray', lw=2,
    ↪ linestyle='--')
axs[0, 0].set_xlabel('Actual Total Medals')
axs[0, 0].set_ylabel('Predicted Total Medals')
axs[0, 0].set_title('Total Medals: Predictions vs Actual')
axs[0, 0].legend()

# Gold Medals
axs[0, 1].scatter(y_test['Gold'], xgb_predictions[:, 1], color='gold',
    ↪ label='Predicted Gold Medals')
axs[0, 1].plot([y_test['Gold'].min(), y_test['Gold'].max()],
    ↪ [y_test['Gold'].min(), y_test['Gold'].max()], color='gray', lw=2,
    ↪ linestyle='--')
axs[0, 1].set_xlabel('Actual Gold Medals')
axs[0, 1].set_ylabel('Predicted Gold Medals')
axs[0, 1].set_title('Gold Medals: Predictions vs Actual')
axs[0, 1].legend()

# Silver Medals
axs[1, 0].scatter(y_test['Silver'], xgb_predictions[:, 2], color='silver',
    ↪ label='Predicted Silver Medals')
axs[1, 0].plot([y_test['Silver'].min(), y_test['Silver'].max()],
    ↪ [y_test['Silver'].min(), y_test['Silver'].max()], color='gray', lw=2,
    ↪ linestyle='--')
axs[1, 0].set_xlabel('Actual Silver Medals')
axs[1, 0].set_ylabel('Predicted Silver Medals')
axs[1, 0].set_title('Silver Medals: Predictions vs Actual')
axs[1, 0].legend()

# Bronze Medals
axs[1, 1].scatter(y_test['Bronze'], xgb_predictions[:, 3], color='#cd7f32',
    ↪ label='Predicted Bronze Medals')
axs[1, 1].plot([y_test['Bronze'].min(), y_test['Bronze'].max()],
    ↪ [y_test['Bronze'].min(), y_test['Bronze'].max()], color='gray', lw=2,
    ↪ linestyle='--')
axs[1, 1].set_xlabel('Actual Bronze Medals')
axs[1, 1].set_ylabel('Predicted Bronze Medals')
axs[1, 1].set_title('Bronze Medals: Predictions vs Actual')
axs[1, 1].legend()

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

Appendix B Code for Great Coach Effect

Input Python source:

```python
import pandas as pd
import chardet
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score



#   CSV
df = pd.read_csv('./2025_Problem_C_Data/summerOly_athletes.csv')

#
us_women_gymnastics = df[(df['NOC'] == 'USA') & ((df['Sport'] == 'Gymnastics'))
    ↪ & (df['Sex'] == 'F')]

# 2020              2024

#         CSV
us_women_gymnastics.to_csv('./2025_Problem_C_Data/us_women_gymnastics.csv',
    ↪ index=False)

#
legendary_years = [1976, 1981, 1984, 1989, 1996, 2000, 2004, 2008, 2012, 2016,
    ↪ 2020]
us_women_gymnastics.loc[:, 'Legendary_Coach'] =
    ↪ us_women_gymnastics['Year'].apply(lambda x: 1 if x in legendary_years else
    ↪ 0)

#
medal_counts = us_women_gymnastics.groupby(['Year',
    ↪ 'Legendary_Coach'])['Medal'].value_counts().unstack(fill_value=0)
medal_counts['Total'] = medal_counts[['Gold', 'Silver', 'Bronze']].sum(axis=1)
medal_counts = medal_counts[['Total', 'Gold', 'Silver', 'Bronze', 'No medal']]
medal_counts = medal_counts.reset_index()

#    medal_countsCSV
medal_counts.to_csv('./2025_Problem_C_Data/medal_counts.csv', index=False)

print(medal_counts)
```

```python
#  programs. csv
programs_df = pd.read_csv('./2025_Problem_C_Data/summerOly_programs.csv')

#   Sports
sports_list = programs_df['Sport'].unique()

#       DataFramemedal_counts
all_medal_counts = pd.DataFrame()

#   Sport
for sport in sports_list:
    #     Sport
    sport_data = df[df['Sport'] == sport]

    #   NOCSex
    grouped_data = sport_data.groupby(['NOC', 'Sex'])

    for (noc, sex), group in grouped_data:
        #
        medal_counts0 =
            ↪ group.groupby('Year')['Medal'].value_counts().unstack(fill_value=0)
        medal_counts0 = medal_counts0.reindex(columns=['Gold', 'Silver',
            ↪ 'Bronze', 'No medal'], fill_value=0)
        medal_counts0['Total'] = medal_counts0[['Gold', 'Silver',
            ↪ 'Bronze']].sum(axis=1)
        medal_counts0 = medal_counts0[['Total', 'Gold', 'Silver', 'Bronze', 'No
            ↪ medal']].reset_index()
        #
        medal_counts0 = medal_counts0.reset_index()

        # #  Sport, NOC,  Sex
        medal_counts0['Sport'] = sport
        medal_counts0['NOC'] = noc
        medal_counts0['Sex'] = sex

        #     medal_countsall_medal_counts
        all_medal_counts = pd.concat([all_medal_counts, medal_counts0],
            ↪ ignore_index=True)

#     medal_countsCSV
all_medal_counts.to_csv('./2025_Problem_C_Data/medal_all_counts.csv',
    ↪ index=False)

print(all_medal_counts)
```

```
# #       medal_countsLegendary_Coach    ,'Year ''Total', 'Gold', 'Silver',
    ↪ 'Bronze', 'No medal       '[2020,13,8,4,1,8]

#
X = medal_counts[['Year', 'Total', 'Gold', 'Silver', 'Bronze', 'No medal']]
y = medal_counts['Legendary_Coach']

#
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    ↪ random_state=42)

#

#
params = {
    'objective': 'reg:squarederror',
    'learning_rate': 0.01,
    'max_depth': 6,
    'min_child_weight': 1,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'n_estimators': 1000,
    'random_state': 42,
    'early_stopping_rounds': 10
}

#
model = xgb.XGBRegressor(**params)
model.fit(X_train, y_train,
         eval_set=[(X_train, y_train), (X_test, y_test)],
         verbose=False)


#

#   Sport, NOC,         Sex
grouped_all_medal_counts = all_medal_counts.groupby(['Sport', 'NOC', 'Sex'])

#       DataFrame
legendary_coach_results = pd.DataFrame()

for (sport, noc, sex), group in grouped_all_medal_counts:
    #
    group['Legendary_Index'] = model.predict(group[['Year', 'Total', 'Gold',
        ↪ 'Silver', 'Bronze', 'No medal']])
```

```
    #  legendary_index >      1
    legendary_threshold = 0.68

    #
    group['Predicted_Legendary_Coach'] = group['Legendary_Index'] >
        ↪ legendary_threshold

    #
    result = group[['Year','Sport', 'NOC', 'Sex', 'Legendary_Index',
        ↪ 'Predicted_Legendary_Coach']]

    #     legendary_coach_results
    legendary_coach_results = pd.concat([legendary_coach_results, result],
        ↪ ignore_index=True)

#     CSV
legendary_coach_results.to_csv('./2025_Problem_C_Data/legendary_coach_results.csv',
    ↪ index=False)

#        CSV
legendary_coach_results_true =
    ↪ legendary_coach_results[legendary_coach_results['Predicted_Legendary_Coach']
    ↪ == True]
legendary_coach_results_true.to_csv('./2025_Problem_C_Data/legendary_coach_results_true.csv',
    ↪ index=False)

print(legendary_coach_results_true)
```

Appendix C Code for Predicting First Medal Countries

Input Python source:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

# Load the data
data = pd.read_csv('./2025_Problem_C_Data/data.csv')


country_mapping = {
    'Soviet Union': 'Russia',
    'West Germany': 'Germany',
    'East Germany': 'Germany',
    'Yugoslavia': 'Serbia',
    'Czechoslovakia': 'Czech Republic',
```

```
    'Bohemia': 'Czech Republic',
    'Russian Empire': 'Russia',
    'United Team of Germany': 'Germany',
    'Unified Team': 'Russia',
    'Serbia and Montenegro': 'Serbia',
    'Netherlands Antilles': 'Netherlands',
    'Virgin Islands': 'United States',
    'ROC': 'Russia',
}


data['NOC'] = data['NOC'].replace(country_mapping)

# Filter out the data for countries that won medals in 1896
data_1896_winners = data[(data['Year'] == 1896) & (data['Total'] >
    ↪ 0)]['NOC'].unique()
data = data[~data['NOC'].isin(data_1896_winners)]

# Create a feature set for each country
country_features = data.groupby('NOC').agg({
    'Year': ['count', 'min', 'max'],
    'Female_Ratio': 'mean',
    'Num_Athletes': 'sum',
    'Num_Sports': 'sum',
    'Num_Events': 'sum',
    'Total': 'sum'
}).reset_index()

country_features.columns = ['NOC', 'Participations', 'First_Year', 'Last_Year',
    ↪ 'Total_Athletes', 'Female_Ratio','Total_Sports', 'Total_Events',
    ↪ 'Total_Medals']

# Add a column for whether the country has won a medal
country_features['Has_Won'] = country_features['Total_Medals'] > 0

# Encode the NOC column
le = LabelEncoder()
country_features['NOC'] = le.fit_transform(country_features['NOC'])

# Split the data into training and testing sets
X = country_features.drop(columns=['Has_Won', 'Total_Medals'])
y = country_features['Has_Won']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    ↪ random_state=42)

# Train a RandomForestClassifier
clf = RandomForestClassifier(n_estimators=10000, random_state=42)
```

```
clf.fit(X_train, y_train)

# Predict the probability of winning a medal for each country
country_features['Win_Probability'] = clf.predict_proba(X)[:, 1]

# Print the countries with the highest probability of winning a medal for the
    ↪ first time
first_time_winners = country_features[country_features['Has_Won'] ==
    ↪ False].sort_values(by='Win_Probability', ascending=False)
# Decode the NOC column back to country names
first_time_winners['NOC'] = le.inverse_transform(first_time_winners['NOC'])
print(first_time_winners[['NOC', 'Win_Probability']].head(10))

# Save the prediction data to a CSV file
country_features.to_csv('./2025_Problem_C_Data/first_win_predictions.csv',
    ↪ index=False)

# Plot a pie chart for the top 10 countries with the highest probability of
    ↪ winning a medal for the first time
import matplotlib.pyplot as plt

top_10_countries = first_time_winners.head(10)
plt.figure(figsize=(10, 7))
plt.pie(top_10_countries['Win_Probability'], labels=top_10_countries['NOC'],
    ↪ autopct='%1.1f%%', startangle=140)
plt.title('Top 10 Countries with Highest Probability of Winning a Medal for the
    ↪ First Time')
plt.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

Appendix D Code for Great Athletes and Key Events

Input Python source:

```
import pandas as pd

#
file_path = "./2025_Problem_C_Data/summerOly_athletes.csv"
data = pd.read_csv(file_path)

#
medal_data = data[data['Medal'] != 'No medal']

#
medal_counts = medal_data.groupby(['NOC', 'Sport',
    ↪ 'Event'])['Medal'].count().reset_index(name='Medal Count')
```

```
#
total_medals_per_country = medal_counts.groupby('NOC')['Medal
    ↪ Count'].sum().reset_index(name='Total Medals')

#              /
medal_data_with_totals = pd.merge(medal_counts, total_medals_per_country,
    ↪ on='NOC')
medal_data_with_totals['Medal Proportion'] = medal_data_with_totals['Medal
    ↪ Count'] / medal_data_with_totals['Total Medals']

#              /
threshold = 0.5 #
reliant_countries = medal_data_with_totals[medal_data_with_totals['Medal
    ↪ Proportion'] > threshold]

#             2/
reliant_countries_top_sports = reliant_countries.groupby('NOC').apply(lambda x:
    ↪ x.nlargest(2, 'Medal Proportion')).reset_index(drop=True)

#
print(reliant_countries_top_sports[['NOC', 'Sport', 'Event', 'Medal
    ↪ Proportion']])
import matplotlib.pyplot as plt

#            /
reliant_countries_top_sports = reliant_countries_top_sports[['NOC', 'Sport',
    ↪ 'Event', 'Medal Proportion']]

#
plt.figure(figsize=(12, 8))
for i, country in enumerate(reliant_countries_top_sports['NOC'].unique()):
    country_data =
        ↪ reliant_countries_top_sports[reliant_countries_top_sports['NOC'] ==
        ↪ country]
    plt.bar(country_data['Sport'] + ' - ' + country_data['Event'],
            country_data['Medal Proportion'],
            label=country, alpha=0.7)

plt.xticks(rotation=90) #       x
plt.xlabel('Sport - Event')
plt.ylabel('Medal Proportion')
plt.title('Countries Relying on One or Two Sports/Events for Most Medals')
plt.legend(title='Country', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

Report on Use of AI

1. OpenAI ChatGPT (Dec 14, 2024 version, ChatGPT-4o)

   Query1: <insert the exact wording you input into the AI tool>

   Output: <insert the complete output from the AI tool>

2. Github Copilot (2024 version, ChatGPT-4o, o1(preview))

   Query1: <insert the exact wording you input into the AI tool>

   Output: <insert the complete output from the AI tool>

3. Deepseek (2025 version)

   Query1: Find some references about Medal prediction in the Olympic Games

   Output: Here are some references related to medal prediction in the Olympic Games, focusing on methodologies, data-driven approaches, and key insights from academic research and practical applications:

---

1. Academic Papers on Medal Prediction

1.1. "Predicting Olympic Medal Counts: The Effects of Economic Development on Olympic Success"

- **Authors**: Daniel K. N. Johnson, Ayfer Ali
- **Journal**: *Social Science Quarterly* (2004)
- **Summary**: This paper explores the relationship between a country's economic development (GDP, population) and its Olympic medal count. It uses regression analysis to predict medal totals based on economic indicators.
- **Key Insight**: Economic factors (GDP per capita, population size) are strong predictors of Olympic success.
- **Link**: DOI:10.1111/j.0038-4941.2004.08501009.x

1.2. "A Machine Learning Approach to Predicting Olympic Medal Counts"

- **Authors**: S. S. R. Abidi, M. Manzoor
- **Conference**: *International Conference on Machine Learning and Applications* (2018)
- **Summary**: This study applies machine learning algorithms (e.g., Random Forest, Gradient Boosting) to predict Olympic medal counts using historical data, economic indicators, and sports-specific features.
- **Key Insight**: Machine learning models outperform traditional regression models in accuracy, especially when incorporating sport-specific data.
- **Link**: IEEE Xplore

1.3. "The Role of Host Nation Advantage in Olympic Success"

- **Authors**: Andrew B. Bernard, Meghan R. Busse
- **Journal**: *The Review of Economics and Statistics* (2004)
- **Summary**: This paper investigates the "host nation advantage" and its impact on medal counts. It uses historical data to quantify the boost in performance for host countries.
- **Key Insight**: Host nations typically win an additional 1.3% of total medals, with the effect lasting for two Olympic cycles.
- **Link**: DOI:10.1162/003465304323023723

---

2. Data-Driven Approaches

2.1. "Gracenote Olympic Medal Predictions"

- **Source**: Gracenote Sports
- **Summary**: Gracenote uses a proprietary algorithm to predict Olympic medal counts based on historical performance, recent results, and athlete rankings. Their predictions are widely cited in the media.
- **Key Insight**: The model incorporates both individual athlete performance and team dynamics.
- **Link**: Gracenote Sports

2.2. "Nielsen's Olympic Performance Predictions"

- **Source**: Nielsen Sports
- **Summary**: Nielsen uses a combination of historical data, athlete rankings, and economic indicators to predict medal counts for each country.
- **Key Insight**: The model highlights the importance of recent performance trends in predicting future success.
- **Link**: Nielsen Sports

---

3. Practical Applications and Case Studies

3.1. "Predicting Olympic Medal Counts Using Bayesian Hierarchical Models"

- **Authors**: Andrew Gelman, Jennifer Hill
- **Book**: *Data Analysis Using Regression and Multilevel/Hierarchical Models* (2006)
- **Summary**: This book includes a case study on predicting Olympic medal counts using Bayesian hierarchical models, which account for country-level and sport-level variability.

- **Key Insight**: Hierarchical models provide more accurate predictions by capturing the nested structure of Olympic data (e.g., athletes within countries, sports within events).
- **Link**: Cambridge University Press

3.2. "The Economics of the Olympics: Predicting Medal Counts"

- **Authors**: Stefan Szymanski
- **Journal**: *Journal of Sports Economics* (2010)
- **Summary**: This paper examines the economic and demographic factors that influence Olympic success, using regression analysis to predict medal counts.
- **Key Insight**: Population size and GDP are the most significant predictors, but sport-specific investments also play a role.
- **Link**: DOI:10.1177/1527002510368101

---

4. Open Data Sources for Medal Prediction

4.1. Olympic Medal Dataset

- **Source**: Kaggle
- **Summary**: A comprehensive dataset containing historical Olympic medal counts, athlete information, and event results.
- **Link**: Kaggle Olympic Dataset

4.2. World Bank Economic Indicators

- **Source**: World Bank
- **Summary**: Provides GDP, population, and other economic indicators that can be used as predictors in medal count models.
- **Link**: World Bank Data

4. Doubao (2025 version)

Query1: The year Bela and Marta instructed the us gym team

Output: Bela and Marta instructed the US gym team in the following years:

- **1981**: They defected to the US and began coaching gymnastics soon after.

- **1984**: They coached Mary Lou Retton, who won America's first gold medal in the Olympic all - arounds.

- **1991**: They trained Kim Zmeskal, who became the world champion.

- **1996**: Marta coached the "Magnificent Seven" member Kerri Strug to win the Olympic team gold.

- **2000**: Bela was named national team coordinator.

- **2001 - 2016**: Marta served as the coordinator of the US women's gymnastics national team.