From Probabilities to Podiums: Who will Shine Bright in LA Olympics 2028?

Team 2525255

January 2025

1 Summary

Olympic Games-"world's foremost sports competitions not just for athletes but also for nation's pride and international recognition.By combining historical data, Mathematical Modeling and socio-economic Factor, This report dive into the complex process of Predicting medal counts for upcoming Los Angeles 2028 Summer Olympics.

By using the provided data , This report develop Statistical model to forecast medal counts for each participating country by Incorporating the external factor such as GDP, Number of participating athlete,host factor and doping incident.By using Poisson distribution and regression techniques,the analysis calculates expected medal counts by incorporating external factors using Poisson regression and error metrics.The results shows significant trends , including the positive hosting effect,as seen with the united states.Emerging nation with increasing GDP and participation rates show potential for their first medal and inclusive growth."Great coach" also give a sudden growth opportunity to some countries.The data driven approach provides actionable insights for Olympic committees to optimize Training programs,allocate resources,and designs strategies for improved performance.The report concludes by highlighting the corelation and intersection of sports,economics and data science,offering a path for future research to refine predictions.

Contents

2 Background

The Olympic Games, the prestige of every country, is held every four years, showcasing athletic excellence and global participation. Dating back to its modern inception in 1896, the Olympics have grown into a premier global sporting event, uniting athletes from diverse cultures and backgrounds [1]. Throughout the Olympic journey, the games have witnessed significant milestones, such as the inclusion of women in 1900 [2], the introduction of the Winter Olympics in 1924 [1], and the expansion to over 300 events in recent years [1].

The Olympics also reflect global socioeconomic and political trends. For instance, the Cold War era saw intense rivalry between the USA and USSR [3], while more recent years have demonstrated the rise of countries like China and Brazil as sporting powerhouses [4]. With advancements in technology, the Games get broadcast live which gives a virtual reality experience and also provides data-driven analytics to engage audiences worldwide [5].

Participation in the Olympics is a strong demonstration of a country's commitment to sports development. Countries invest heavily in infrastructure, athlete training, and international collaborations to improve their performance. Analyzing these trends in participation and medal distribution helps uncover patterns, assess competitiveness, and guide future preparation for the games.

The objective of this report is to examine Olympic data from 1896 to 2024, focusing on the number of participants per country and medal counts. Using statistical methods, particularly Poisson probability and distribution, this report aims to predict medal counts for each event in the upcoming Olympics as well as total medals per country. This analysis provides insights for national sports committees, and athletes to allocate resources strategically.

3 Assumptions

3.1 Statistical Assumptions

- Medal Distribution follows a Poisson process, where the occurrence of medals in an event is modeled using a predictable average rate.
- Each event is treated as an independent trial, meaning the outcome of one event does not influence others.

3.2 Socio-Economic Assumptions

- GDP is directly proportional to a country's performance in the Olympics, as wealthier nations invest more in invest more in sports infrastructure, athlete training, and technology.
- Countries with larger populations tend to have higher participation rates, increasing their chances of winning medals.

3.3 Data Assumptions

- Historical data from 1896-2024 is representative of future trends, assuming no major disruption or rule changes in the Olympic Games.
- External factors such as geopolitical conflicts, pandemics, or other anomalies have minimal long-term impact on the trends analyzed.

3.4 Event-Specific Assumptions

- Performance is each event is influenced primarily by training, skill, and preparation rather than external factors like weather, or doping.
- Medal probabilities are constant for a country in a specific event over time unless there are significant changes in sports regulations or participation.

3.5 Additional Assumption

- The likelihood of a country winning a medal increases proportionally with prior historical performance in the event.
- Host countries may show slight advantages due to familiarity with venues and support from local audiences, though this is not explicitly modeled.

3.6 Model-Specific Assumptions

- Poisson distribution accurately captures the discrete and rare nature of medal-winning events.
- Factors like GDP, population, and historical performance are key variables influencing medal counts and are sufficient for prediction without considering additional variables like weather or political stability.

4 Statistical Approach

The Best approach to find the future predictions by using Historical Data is by using Probabilities,To choose best Distribution applicable here . first have to find Parameters and these are:
1. Number of medals
2. Events

Since,

- The number of medals won by a specific country in a specific event is count-based outcome.
- Medal counts are independent across events and countries and at a continuous rate.

Another question that is asked, is whether or not a coach can truly have any significant affect on a team in the Olympics success. Using the data provided a Chi- Squared Test can be performed in order to see if there is any statistical significance in the medal count while the coach was a part of the team for the Olympics and the medal count once the coach has left the Olympic team to retire or join another country.

Therefore, Poisson Distribution is Appropriate Approach.

4.1 Key Formula and Parameters

The Poisson Probability function is given by :

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Where:

- **X**: The number of medals predicted for a country in a specific event.
- k: The observed or expected number of medals (e.g., 0, 1, 2, etc.).
- λ: The average rate (mean) of medals won, calculated from historical data.

The Expected value of the Poisson distribution is given by:

$$E(X) = n \cdot P$$

- n: Number of medals won.
- P: The probability of winning a medal in each event.

The Chi-Squared test is given by:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Where:

- χ: The Chi squared test statistic.
- O_i: The observed values.
- E_i: The expected values.

4.2 Data Preparation

Step 1.To calculate the probability, λ is required.

The parameter λ for a Poisson distribution, representing the expected value or mean number of medals, can be calculated as:

$$\lambda = \frac{\text{Total Number of Medals}}{\text{Total Number of Participants}}$$

4.2.1

Processing the Athlete Dataset for Total Participants

To calculate the total number of participants for each country (*Team*) in each year, the following steps were undertaken:

1. **Renaming Columns:** The dataset 'summer_oly_athlete' initially had a column labeled `NOC`, representing the National Olympic Committee codes. To align with other datasets that used the term `NOC` instead of `Team`, the column was renamed from `Team` to `NOC`. This ensures consistency across datasets and facilitates future merging operations.
2. **Grouping and Arranging Data:** After renaming, the dataset was grouped by `NOC` (country) and `Year`. For each group, the total number of athletes (participants) was calculated. This count reflects the number of athletes representing each country in a specific Olympic year.
3. **Calculating Total Participants:** For every country (`NOC`) and year, the total participants were obtained by counting the rows within each group. Each row in the dataset represents an individual athlete, so summing these rows per group provides the required participant totals.
4. **Resulting Dataset:** The final output was a dataset with the following structure:
   - **NOC:** The country or National Olympic Committee name.
   - **Year:** The year of the Olympic event.
   - **Total Participants:** The total number of athletes representing the country in that year.

   This dataset provides a clear and concise summary of athlete participation across countries and years.

NOC	Year	Total Participants
30. Februa	1952	2
A North An	1900	4
AIN	2024	46
Acipactli	1964	3
Acturus	1948	2
Afghanista	1936	16
Afghanista	1948	25
Afghanista	1956	12
Afghanista	1960	16
Afghanista	1964	8
Afghanista	1968	5
Afghanista	1972	8
Afghanista	1980	11
Afghanista	1988	5
Afghanista	1996	2
Afghanista	2004	5
Afghanista	2008	4
Afghanista	2012	6
Afghanista	2016	3

Figure 1: Snippet of the dataset with columns for NOC, Year, and Total Participants.

Step 2: Assorted each country by each event and each medal type .

Year	NOC	Sport	Event	Bronze	Gold	No medal	Silver
1896	Australia	Athletics	Athletics M	0	1	0	0
1896	Australia	Athletics	Athletics M	0	1	0	0
1896	Australia	Athletics	Athletics M	0	0	1	0
1896	Australia	Tennis	Tennis Mer	0	0	1	0
1896	Australia/(	Tennis	Tennis Mer	2	0	0	0
1896	Austria	Cycling	Cycling Me	1	0	0	0
1896	Austria	Cycling	Cycling Me	0	0	1	0
1896	Austria	Cycling	Cycling Me	0	1	0	0
1896	Austria	Cycling	Cycling Me	1	0	0	0
1896	Austria	Fencing	Fencing Me	0	0	1	0
1896	Austria	Swimming	Swimming	0	0	1	0
1896	Austria	Swimming	Swimming	0	0	0	1
1896	Austria	Swimming	Swimming	0	1	0	0
1896	Denmark	Athletics	Athletics M	0	0	1	0
1896	Denmark	Athletics	Athletics M	0	0	2	0
1896	Denmark	Athletics	Athletics M	0	0	1	0

Figure 2: Each country is assorted by number of each medal type

Step 3. Merge λ for each type of medal and number of each type of medal each country each year.

Year	NOC	Sport	Event	Bronze_x	Gold_x	No medal	Silver_x	Rank	Gold_y	Silver_y	Bronze_y	Total	Total Partic	Lambda_G	Lambda_Si	Lambda_Bronze
1896	Australia	Athletics	Athletics M	0	1	0	0	8	2	0	0	2	4	0.5	0	0
1896	Australia	Athletics	Athletics M	0	1	0	0	8	2	0	0	2	4	0.5	0	0
1896	Australia	Athletics	Athletics M	0	0	1	0	8	2	0	0	2	4	0.5	0	0
1896	Australia	Tennis	Tennis Mer	0	0	1	0	8	2	0	0	2	4	0.5	0	0
1896	Austria	Cycling	Cycling Me	1	0	0	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Cycling	Cycling Me	0	0	1	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Cycling	Cycling Me	0	1	0	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Cycling	Cycling Me	1	0	0	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Fencing	Fencing Me	0	0	1	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Swimming	Swimming	0	0	1	0	7	2	1	2	5	8	0.25	0.125	0.25
1896	Austria	Swimming	Swimming	0	0	0	1	7	2	1	2	5	8	0.25	0.125	0.25

Figure 3: Merged data for lambda and number of medal for each medal type

Step 4 . Calculating Probability (wining a type of medal in particular country in specific year bya country)

$$P((medaltype(b)bycountry(i)inevent(t)bycountry(x)) = \frac{\lambda_b^k e^{-\lambda_b}}{k!}$$

Where k is Number of medals

NOC	Event	Mean P(Gc	Mean P(Sil	Mean P(Bronze)
Afghanista	Athletics M	0	0	0.812641
Afghanista	Athletics W	0	0	0.812641
Afghanista	Boxing Mer	0	0	0.846482
Afghanista	Judo Men's	0	0	0.846482
Afghanista	Taekwondo	0	0	0.459941
Afghanista	Taekwondo	0	0	0.1947
Afghanista	Taekwondo	0	0	0.846482
Albania	10m Air Pis	0	0	0.800737
Albania	25m Pistol	0	0	0.800737
Albania	Men's 100r	0	0	0.800737
Albania	Men's 100r	0	0	0.800737
Albania	Men's Free	0	0	0.800737
Albania	Men's Free	0	0	0.177942
Albania	Men's Free	0	0	0.177942
Albania	Women's 2	0	0	0.800737
Albania	Women's 3	0	0	0.800737
Algeria	10m Air Pis	0.96429	0	0.981982

Figure 4: Probability of each type of medals, for each event and each country

Step 5. Finding Expected number of Medal

$$E((medaltype(b)bycountry(i)inevent(t)bycountry(x) = numnerofmedaltype(b)woninspecificyear \cdot Me$$

	Gold_x	Silver_x	Bronze_x	Lambda_G	Lambda_S	Lambda_B	Mean P(Gc	Mean P(Sil	Mean P(Br	Expected_(	Expected_S	Expected_I	Total_Expected_Medals
M	1	0	0	0.5	0	0	0.821422	0.861767	0.862572	1	0	0	1
M	0	0	0	0.05	0.016667	0.033333	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.034483	0.068966	0.034483	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0	0	0.025641	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.044444	0.014815	0.022222	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	1	0.032338	0.0199	0.034826	0.821422	0.861767	0.862572	0	0	1	1
M	0	0	0	0.015707	0.005236	0.026178	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.022321	0.03125	0.022321	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.030075	0.026316	0.007519	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0	0.00365	0.014599	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.011869	0.023739	0.035608	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.00885	0.017699	0.014749	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.016605	0.016605	0.042435	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.020997	0.032808	0.02231	0.821422	0.861767	0.862572	0	0	0	0
M	0	0	0	0.025045	0.026834	0.030411	0.821422	0.861767	0.862572	0	0	0	0

Figure 5: Expected number of number of medal for each year for each country in each event and each year

Step 6. Adding up total medal for each country for each year

Step 7. Comparison of Actual (Given Dataset) vs. Predicted Gold Medals by using correlation coefficient.

The **correlation coefficient** (r) is a numerical value that quantifies the strength and direction of a linear relationship between two variables.

Range of r:

$$-1 \leq r \leq 1$$

- $r = 1$: Perfect positive correlation.
- $r = -1$: Perfect negative correlation.
- $r = 0$: No linear correlation.

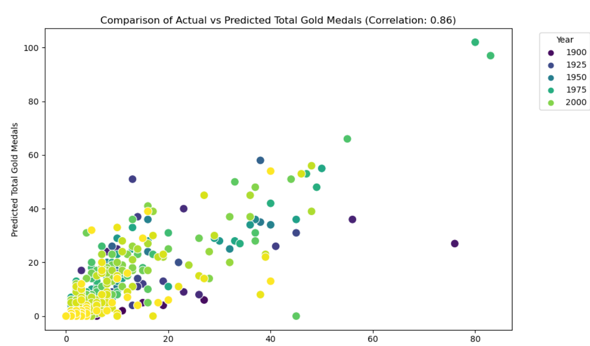

Figure 6: Correlation of gold medals

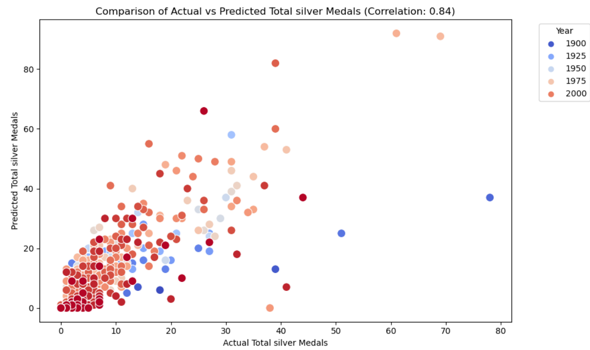

Figure 7: Correlation of Silver medals

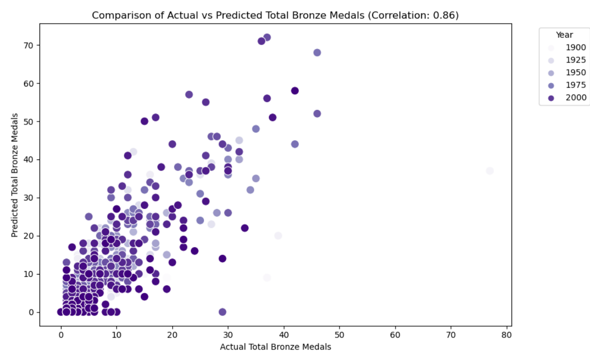

Figure 8: Correlation of Bronze medals

4.2.2 Key Observations

- A correlation coefficient of 0.86 suggests that the prediction model performs well in forecasting medal counts.
- Points closer to the diagonal line represent countries where the predictions closely match the actual values.
- Points farther away from the diagonal indicate prediction errors, but these are relatively few.

4.2.3 Coaching Factor

To fully understand if coaching really does affect whether or not athletes at the Olympic games win more with a highly regarded coach, then there is no better coach to analyze from the dataset then the performances of swimmers from 2004 to 2016, as they were coached by the prolific Bob Bowman. The method that is used in order to see if coach Bowman has an affect on these athletes is to set up an A/B test using the Chi-Square Test, with two hypothesis'.

- H_0 : Bob Bowman does not affect winning

- H_1 : Bob Bowman does affect winning

The Chi-Sqaure test is going to be conducted on the medal counts between the years of 2004 to 2016, and on the years after Bowman left the team in 2020-2024, using python with a significance level of 0.05. If the p-value produced in the Chi-Square test is less than 0.05, then the null hypothesis is rejected and the alternative hypothesis can be accepted. The data was prepared through the use of the pandas library, the code can be found in the code section the report, that was used to filter out the data needed.

4.3 Results

Finally calculating Predictions for 2028 medal outcomes for each country by using 2024 medal outcome , calculated Lembda for each type of medal and add medal count by each country.

$$E((medaltype(b)bycountry(i)inevent(t)bycountry(x) = numnerofmedaltype(b)woninspecificyear\cdot Me$$

	NOC	Expected_Gold_2028	Expected_Silver_2028	Expected_Bronze_2028	Expected_Total_Medals_2028
0	Albania	0.000000	0.000000	0.000000	0.0
1	Algeria	0.000000	0.000000	0.000000	0.0
2	Argentina	0.000000	0.000000	0.000000	0.0
3	Armenia	0.000000	0.035185	0.000000	0.0
4	Australia	1.333221	5.242694	3.362722	10.0
...	...	...	...	...	...
77	Uganda	0.000000	0.011840	0.000000	0.0
78	Ukraine	0.073980	0.000000	0.000000	0.0
79	United States	63.715909	29.914009	42.947501	137.0
80	Uzbekistan	0.103250	0.000000	0.201821	0.0
81	Zambia	0.000000	0.000000	0.000000	0.0

Figure 9: Predictions for 2028 medal count

From the Chi-square test conducted in python it was quite evident that the null hypothesis could be rejected and there is a clear difference in the results of the team USA's swimming performance as they did end up with more medals period with Bowman being there as their coach. The signifiance level chosen could have been even lower than 0.05 and it and it still would have been wise for the null hypothesis to be rejected, it is with that we can conclude that Bowman may in fact be very important to the structure of team USA and that it may be wise for him to join the coaching staff again in 2028 as he can help his country to win more medals, not just gold medals but clearly just by looking at the data overall.

With a p-value of below even 0.01, it is quite clear that coach Bowman does make a statistical impact on the teams winning in the Olympics. The results are from the python code that is found in section 6 of the report.

```
Chi-square statistic: 19.981506759538213
P-value: 0.00017124696991398705
Degrees of freedom: 3
Expected frequencies:
[[127.83443709  71.16556291]
 [ 83.50993377  46.49006623]
 [ 45.60927152  25.39072848]
 [131.04635762  72.95364238]]
Reject H₀: There is a statistically significant difference.
```

Figure 10: Chi-Square Test Results

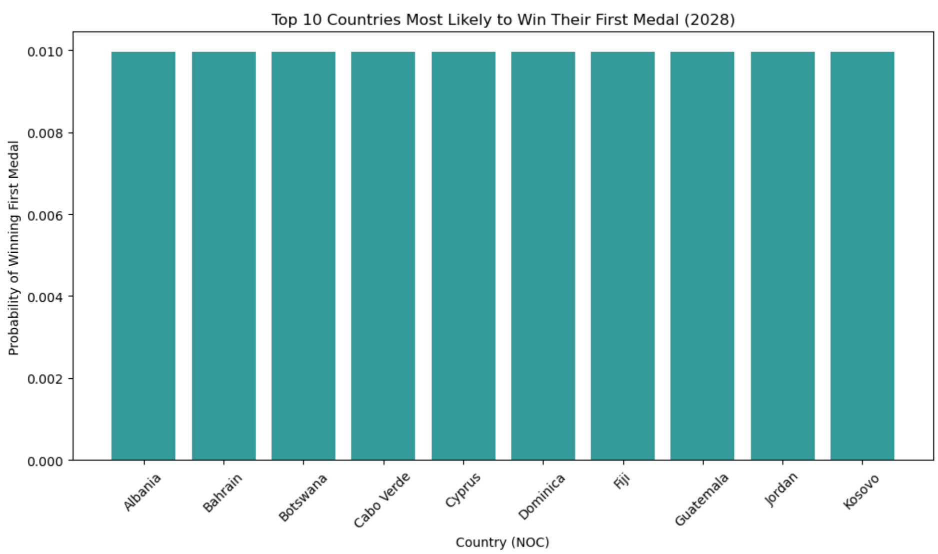

Figure 11: Country most likely to get their first Medal

5 Code

Model Evaluation code

6 Model Evaluation Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import zipfile
import pandas as pd
zip_file_path = '2025_Problem_C_Data.zip'
with zipfile.ZipFile(zip_file_path, 'r') as z:
    z.extractall('/mnt/data/extracted_athletes_data')
    extracted_files = z.namelist()
extracted_files


# In[15]:


pip install Pygments


# In[2]:

```

```python

# Path to the athletes file
athletes_file_path =
     '/mnt/data/extracted_athletes_data/2025_Problem_C_Data/summerOly_athletes.csv'

# Load the data into a pandas DataFrame
athletes_df = pd.read_csv(athletes_file_path)

# Display the first few rows to inspect the data
print(athletes_df.head())

unique_noc_teams = athletes_df[['NOC',
     'Team']].drop_duplicates()

# Saving the extracted data as a CSV file
output_file_path = "noc_team_mappings.csv"
unique_noc_teams.to_csv(output_file_path,
     index=False)

# Displaying the saved file path
output_file_path


# ##### Changing "Team" column name by NOC to
     maintain similarity among another Data Sets

# In[3]:


athletes_file_path =
     '/mnt/data/extracted_athletes_data/2025_Problem_C_Data/summerOly_athletes.csv'
athletes_df = pd.read_csv(athletes_file_path)

# Rename 'team' column to 'noc' and 'noc' column
     to 'team'
athletes_df.rename(columns={'Team': 'NOC', 'NOC':
     'Team'}, inplace=True)
athletes_df.drop(columns=['Team'], inplace=True)
output_file_path =
     'modified_summerOly_athletes.csv'
athletes_df.to_csv(output_file_path, index=False)

print(f"Modified dataset saved to:
     {output_file_path}")


# ##### Grouping the data set country wise for
     each year for each athlete

# In[4]:


# Group the data by country ( NOC)
grouped_data = athletes_df.groupby('NOC')



country_summary =
     grouped_data.size().reset_index(name='Total
     Athletes')

if 'Medal' in athletes_df.columns:
    medal_summary = athletes_df.groupby(['NOC',
        'Medal','Year']).size().unstack(fill_value=0)
    country_summary = pd.merge(country_summary,
        medal_summary, on='NOC', how='left')

output_file = 'country_wise_summary.csv'
country_summary.to_csv(output_file, index=False)

print(f"Country-wise data has been saved to
     {output_file}")


# In[5]:

# Group the data by 'Team' and 'Year'
participants_summary =
     athletes_df.groupby(['NOC','Year']).size().reset_index(name='Total
     Participants')

# Save the country-year-wise data to a CSV
output_file =
     'participants_per_country_per_year.csv'
participants_summary.to_csv(output_file,
     index=False)

print(f"Participants per country per year have
     been saved to {output_file}")


# #### Mergeing two Data Sets

# In[6]:


# Load the Medal Counts and Participants data
medal_counts_file_path =
     '/mnt/data/extracted_athletes_data/2025_Problem_C_Data/summerOly_me
participants_file_path =
     'participants_per_country_per_year.csv'

# Load the data into DataFrames
medal_counts_df =
     pd.read_csv(medal_counts_file_path)
participants_df =
     pd.read_csv(participants_file_path)

# Merge the two datasets on 'Team' and 'Year'
merged_data = pd.merge(medal_counts_df,
     participants_df, on=['NOC', 'Year'],
     how='left')

# Save the merged data to a new CSV file
output_file_path =
     'merged_medal_participants_data.csv'
merged_data.to_csv(output_file_path, index=False)



# In[7]:


# Load the CSV file
file_path = 'merged_medal_participants_data.csv'
df = pd.read_csv(file_path)


# Calculate lambda () for Gold, Silver, and Bronze
     medals
df['Lambda_Gold'] = df['Gold'] / df['Total
     Participants']
df['Lambda_Silver'] = df['Silver'] / df['Total
     Participants']
df['Lambda_Bronze'] = df['Bronze'] / df['Total
     Participants']

# Save the results to a new CSV file
output_file =
     'calculated_lambda_expected_medals.csv'
df.to_csv(output_file, index=False)

print(f"Calculation completed. Results saved to
     {output_file}.")

```

```python

# In[8]:


# Load the athletes data
output_file_path = 'modified_summerOly_athletes.csv'
athletes_df = pd.read_csv(output_file_path)

# Group data by Year, Team, Sport, and Event to count total medals by type
event_medal_summary = athletes_df.groupby(['Year', 'NOC', 'Sport', 'Event', 'Medal']).size().reset_index(name='Medal Count')

# Pivot the table to display medal counts for each type (Gold, Silver, Bronze, No medal) side by side
event_medal_summary_pivot = event_medal_summary.pivot_table(
    index=['Year', 'NOC', 'Sport', 'Event'],
    columns='Medal',
    values='Medal Count',
    fill_value=0
).reset_index()

# Save the results to a CSV file
output_file = 'Assorted_by_number_medals.csv'
event_medal_summary_pivot.to_csv(output_file, index=False)

print(f"Results saved to {output_file}")


# In[9]:


# Update the file paths with the actual locations of your files
assorted_file_path = 'Assorted_by_number_medals.csv'
lambda_file_path = 'calculated_lambda_expected_medals.csv'

# Load the CSV files
assorted_data = pd.read_csv(assorted_file_path)
lambda_data = pd.read_csv(lambda_file_path)

# Merge the datasets on common columns (e.g., NOC and Year)
merged_data = pd.merge(
    assorted_data,
    lambda_data,
    how='inner',
    left_on=['Year', 'NOC'],
    right_on=['Year', 'NOC']
)

# Save the merged dataset to a new CSV file
merged_file_path = 'merged_medals_lambda_data.csv'
merged_data.to_csv(merged_file_path, index=False)

print(f"Merged data saved to {merged_file_path}.")


# In[10]:


# Update the path to the actual location of the file
lambda_file_path ='merged_medals_lambda_data.csv'

# Reload the lambda data file
lambda_data = pd.read_csv(lambda_file_path)

# Select the required columns
selected_columns = ['Rank', 'NOC','Year','Event', 'Gold_x', 'Silver_x', 'Bronze_x', 'Lambda_Gold', 'Lambda_Silver', 'Lambda_Bronze']
filtered_data = lambda_data[selected_columns]

# Save the filtered data to a new CSV file
filtered_file_path = 'filtered_medal_lambda_data.csv'
filtered_data.to_csv(filtered_file_path, index=False)

print(f"Filtered data with selected columns saved to {filtered_file_path}.")


# In[11]:


from scipy.stats import poisson

# Reload the filtered data file
probabilities_file_path = 'filtered_medal_lambda_data.csv'
filtered_data = pd.read_csv(probabilities_file_path)

# Ensure probabilities are correctly calculated and do not exceed 1
def calculate_probability(count, lambda_value):
    if lambda_value == 0:
        return 0
    prob = poisson.pmf(count, lambda_value)
    return min(prob, 1) # Ensure probability does not exceed 1


# Add probability columns for Gold, Silver, and Bronze
filtered_data['P(Gold)'] = filtered_data.apply(lambda x: calculate_probability(x['Gold_x'], x['Lambda_Gold']), axis=1)
filtered_data['P(Silver)'] = filtered_data.apply(lambda x: calculate_probability(x['Silver_x'], x['Lambda_Silver']), axis=1)
filtered_data['P(Bronze)'] = filtered_data.apply(lambda x: calculate_probability(x['Bronze_x'],x['Lambda_Bronze']), axis=1)

# Group by NOC and Event to calculate the mean probabilities for Gold, Silver, and Bronze
mean_probabilities = filtered_data.groupby(['NOC', 'Event']).agg({
    'P(Gold)': 'mean',
    'P(Silver)': 'mean',
    'P(Bronze)': 'mean',

}).reset_index()

# Rename the columns for clarity
mean_probabilities.rename(columns={
    'P(Gold)': 'Mean P(Gold)',
    'P(Silver)': 'Mean P(Silver)',
    'P(Bronze)': 'Mean P(Bronze)',
```

```
}, inplace=True)

# Save the results to a new CSV file
mean_probabilities_file =
     'mean_probabilities_per_event.csv'
mean_probabilities.to_csv(mean_probabilities_file,
     index=False)

print(f"Mean probabilities for Gold, Silver, and
     Bronze saved to {mean_probabilities_file}.")


# In[12]:


# Load the datasets
filtered_medal_lambda_df =
     pd.read_csv('filtered_medal_lambda_data.csv')
mean_probabilities_df =
     pd.read_csv('mean_probabilities_per_event.csv')

# Ensure consistent data types
filtered_medal_lambda_df["NOC"] =
     filtered_medal_lambda_df["NOC"].astype(str)
filtered_medal_lambda_df["Event"] =
     filtered_medal_lambda_df["Event"].astype(str)
mean_probabilities_df["NOC"] =
     mean_probabilities_df["NOC"].astype(str)
mean_probabilities_df["Event"] =
     mean_probabilities_df["Event"].astype(str)

# Merge both datasets on NOC and Event
merged_df = pd.merge(filtered_medal_lambda_df,
     mean_probabilities_df, on=["NOC", "Event"],
     how="inner")

# Calculate the expected values
merged_df["Expected_Gold"] = (merged_df["Gold_x"]
     * merged_df["Mean P(Gold)"]).round()
merged_df["Expected_Silver"] =
     (merged_df["Silver_x"] * merged_df["Mean
     P(Silver)"]).round()
merged_df["Expected_Bronze"] =
     (merged_df["Bronze_x"] * merged_df["Mean
     P(Bronze)"]).round()

# Add a total expected medals column
merged_df["Total_Expected_Medals"] = (
    merged_df["Expected_Gold"] +
        merged_df["Expected_Silver"] +
        merged_df["Expected_Bronze"]
)

# Group by Year and NOC for country-wise totals
country_year_totals = merged_df.groupby(["Year",
     "NOC"]).agg(
    Total_Gold=("Expected_Gold", "sum"),
    Total_Silver=("Expected_Silver", "sum"),
    Total_Bronze=("Expected_Bronze", "sum"),
    Total_Medals=("Total_Expected_Medals", "sum")
).reset_index()

# Save the detailed results
merged_df.to_csv('detailed_expected_medals_output.csv',
     index=False)

# Save the country-wise totals
country_year_totals.to_csv('country_year_totals.csv',
     index=False)

# Display the results
print(country_year_totals)


# In[13]:


import matplotlib.pyplot as plt
import seaborn as sns
import os

# Load the actual medal counts from the extracted
     folder
actual_medals_file = medal_counts_file_path
actual_medals_df = pd.read_csv(actual_medals_file)

# Load the predicted medal counts file
predicted_medals_file = 'country_year_totals.csv'
predicted_medals_df =
     pd.read_csv(predicted_medals_file)

# Merge the actual and predicted data on Year and
     NOC
comparison_df = pd.merge(
    actual_medals_df,
    predicted_medals_df,
    on=["Year", "NOC"],
    suffixes=('_Actual', '_Predicted')
)

# Calculate correlation between actual and
     predicted medals
correlation = comparison_df[["Gold",
     "Total_Gold"]].corr().iloc[0, 1]

# Plot the comparison
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=comparison_df,
    x="Gold",
    y="Total_Gold",
    hue="Year",
    palette="viridis",
    s=100
)
plt.title(f"Comparison of Actual vs Predicted
     Total Gold Medals (Correlation:
     {correlation:.2f})")
plt.xlabel("Actual Total Gold Medals")
plt.ylabel("Predicted Total Gold Medals")
plt.legend(title="Year", bbox_to_anchor=(1.05, 1),
     loc='upper left')
plt.tight_layout()
plt.show()


# In[14]:


# Calculate correlation between actual and
     predicted medals
correlation = comparison_df[["Silver",
     "Total_Silver"]].corr().iloc[0, 1]

# Plot the comparison
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=comparison_df,
    x="Silver",
    y="Total_Silver",
    hue="Year",
    palette="coolwarm", # Change to a different
        palette (e.g., "coolwarm", "viridis",
        etc.)
```

```python
    s=100

)
plt.title(f"Comparison of Actual vs Predicted Total silver Medals (Correlation: {correlation:.2f})")
plt.xlabel("Actual Total silver Medals")
plt.ylabel("Predicted Total silver Medals")
plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()


# In[15]:


# Calculate correlation between actual and predicted medals
correlation = comparison_df[["Bronze", "Total_Bronze"]].corr().iloc[0, 1]

# Plot the comparison
plt.figure(figsize=(10, 6))
sns.scatterplot(
    data=comparison_df,
    x="Bronze",
    y="Total_Bronze",
    hue="Year",
    palette="Purples", # Change to a different palette (e.g., "coolwarm", "viridis", etc.)
    s=100

)
plt.title(f"Comparison of Actual vs Predicted Total Bronze Medals (Correlation: {correlation:.2f})")
plt.xlabel("Actual Total Bronze Medals")
plt.ylabel("Predicted Total Bronze Medals")
plt.legend(title="Year", bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()


# In[17]:


# Load the total medal counts for 2024 and the lambda data
total_medals_2024_file = 'country_year_totals.csv'
filtered_medal_lambda_file = 'filtered_medal_lambda_data.csv'

# Load the datasets
total_medals_2024_df = pd.read_csv(total_medals_2024_file)
filtered_medal_lambda_df = pd.read_csv(filtered_medal_lambda_file)

# Filter for the year 2024
total_medals_2024 = total_medals_2024_df[total_medals_2024_df["Year"] == 2024]

# Calculate total medals for all countries in each category (gold, silver, bronze)
total_gold_2024 = total_medals_2024["Total_Gold"].sum()
total_silver_2024 = total_medals_2024["Total_Silver"].sum()
total_bronze_2024 = total_medals_2024["Total_Bronze"].sum()

# Calculate probabilities for each medal type
total_medals_2024["Gold_Probability_2028"] = total_medals_2024["Total_Gold"] / total_gold_2024
total_medals_2024["Silver_Probability_2028"] = total_medals_2024["Total_Silver"] / total_silver_2024
total_medals_2024["Bronze_Probability_2028"] = total_medals_2024["Total_Bronze"] / total_bronze_2024

# Aggregate lambda values for gold, silver, and bronze
lambda_totals = filtered_medal_lambda_df.groupby("NOC").agg({
   "Lambda_Gold": "sum",
   "Lambda_Silver": "sum",
   "Lambda_Bronze": "sum"
}).reset_index()

# Merge probabilities with lambda totals
merged_df = pd.merge(total_medals_2024, lambda_totals, on="NOC", how="inner")
# Predict medals for each type using the formula: Expected_Medals = Probability * Lambda
merged_df["Expected_Gold_2028"] = merged_df["Gold_Probability_2028"] * merged_df["Lambda_Gold"]
merged_df["Expected_Silver_2028"] = merged_df["Silver_Probability_2028"] * merged_df["Lambda_Silver"]
merged_df["Expected_Bronze_2028"] = merged_df["Bronze_Probability_2028"] * merged_df["Lambda_Bronze"]

# Calculate total predicted medals for 2028
merged_df["Expected_Total_Medals_2028"] = (
   merged_df["Expected_Gold_2028"] +
   merged_df["Expected_Silver_2028"] +
   merged_df["Expected_Bronze_2028"]
).round()

# Save the predictions
output_file = 'predicted_medals_2028.csv'
merged_df[["NOC", "Expected_Gold_2028", "Expected_Silver_2028", "Expected_Bronze_2028", "Expected_Total_Medals_2028"]].to_csv(output_file, index=False)

# Display the result
merged_df[["NOC", "Expected_Gold_2028", "Expected_Silver_2028", "Expected_Bronze_2028", "Expected_Total_Medals_2028"]]


# In[ ]:




# In[ ]:




# In[18]:
```

```python

import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

# Load the datasets
total_medals_2024_file = 'country_year_totals.csv'
filtered_medal_lambda_file =
    'filtered_medal_lambda_data.csv'

# Load the data
total_medals_2024_df =
    pd.read_csv(total_medals_2024_file)
filtered_medal_lambda_df =
    pd.read_csv(filtered_medal_lambda_file)

# Filter for the year 2024
total_medals_2024 =
    total_medals_2024_df[total_medals_2024_df["Year"]
    == 2024].copy()

# Calculate total medals for all countries in each
    category (gold, silver, bronze)
total_gold_2024 =
    total_medals_2024["Total_Gold"].sum()
total_silver_2024 =
    total_medals_2024["Total_Silver"].sum()
total_bronze_2024 =
    total_medals_2024["Total_Bronze"].sum()

# Calculate probabilities for each medal type
total_medals_2024["Gold_Probability_2028"] =
    total_medals_2024["Total_Gold"] /
    total_gold_2024
total_medals_2024["Silver_Probability_2028"] =
    total_medals_2024["Total_Silver"] /
    total_silver_2024
total_medals_2024["Bronze_Probability_2028"] =
    total_medals_2024["Total_Bronze"] /
    total_bronze_2024

# Predict expected medals for 2028 based on 2024
    data
total_medals_2024["Expected_Gold_2028"] =
    total_medals_2024["Total_Gold"] *
    total_medals_2024["Gold_Probability_2028"]
total_medals_2024["Expected_Silver_2028"] =
    total_medals_2024["Total_Silver"] *
    total_medals_2024["Silver_Probability_2028"]
total_medals_2024["Expected_Bronze_2028"] =
    total_medals_2024["Total_Bronze"] *
    total_medals_2024["Bronze_Probability_2028"]

# Total predicted medals for 2028
total_medals_2024["Expected_Total_Medals_2028"] = (
    total_medals_2024["Expected_Gold_2028"] +
    total_medals_2024["Expected_Silver_2028"] +
    total_medals_2024["Expected_Bronze_2028"]
).round()

# Identify first-time medal-winning countries
historical_medals =
    total_medals_2024_df.groupby("NOC")["Total_Medals"].sum().reset_index()
first_medal_candidates =
    historical_medals[historical_medals["Total_Medals"]
    == 0]
first_medal_candidates =
    first_medal_candidates.merge(total_medals_2024,
    on="NOC", how="left")

# Adjust zero probabilities with a small base value
first_medal_candidates["Expected_Total_Medals_2028"]
    =
    first_medal_candidates["Expected_Total_Medals_2028"].replace(0,
    0.01)

# Calculate probability of winning at least one
    medal
first_medal_candidates["First_Medal_Probability"]
    = 1 -
    np.exp(-first_medal_candidates["Expected_Total_Medals_2028"])

# Evaluate the model accuracy
actual_medals_2024 =
    total_medals_2024["Total_Medals"]
predicted_medals_2024 =
    total_medals_2024["Expected_Total_Medals_2028"]
mse = mean_squared_error(actual_medals_2024,
    predicted_medals_2024)
correlation = np.corrcoef(actual_medals_2024,
    predicted_medals_2024)[0, 1]

# Visualization: Actual vs. Predicted Medals
plt.figure(figsize=(10, 6))
plt.scatter(actual_medals_2024,
    predicted_medals_2024, alpha=0.7,
    c=total_medals_2024["Year"], cmap="viridis")
plt.colorbar(label="Year")
plt.title(f"Comparison of Actual vs Predicted
    Total Medals (Correlation:
    {correlation:.2f})")
plt.xlabel("Actual Total Medals (2024)")
plt.ylabel("Predicted Total Medals (2028)")
plt.grid()
plt.show()

# Visualization: First Medal Probabilities
plt.figure(figsize=(12, 6))
first_medal_candidates =
    first_medal_candidates.sort_values("First_Medal_Probability",
    ascending=False).head(10)
plt.bar(first_medal_candidates["NOC"],
    first_medal_candidates["First_Medal_Probability"],
    color="teal", alpha=0.8)
plt.title("Top 10 Countries Most Likely to Win
    Their First Medal (2028)")
plt.xlabel("Country (NOC)")
plt.ylabel("Probability of Winning First Medal")
plt.xticks(rotation=45)
plt.show()

# Save predictions and first-time medal
    probabilities to CSV
output_file_1 = 'predicted_medals_2028.csv'
output_file_2 = 'first_medal_candidates.csv'
total_medals_2024[[
    "NOC", "Expected_Gold_2028",
        "Expected_Silver_2028",
        "Expected_Bronze_2028",
        "Expected_Total_Medals_2028"
]].to_csv(output_file_1, index=False)
first_medal_candidates[[
    "NOC", "First_Medal_Probability"
]].to_csv(output_file_2, index=False)
```

Listing 1: Data Filteration and Results

```python
import pandas as pd
from scipy.stats import chi2_contingency

df1 = pd.read_csv("summerOly_athletes.csv")


bowman_years = df1[(df1["Sport"] == "Swimming") &
     (df1["Year"].between(2004,2016)) &
     (df1["NOC"] == "USA")]

print(bowman_years["Medal"].value_counts())

non_bowman_years_before = df1[(df1["Sport"] ==
     "Swimming") & (df1["Year"] <= 2000) &
     (df1["NOC"] == "USA")]
counts_before =
     non_bowman_years_before["Medal"].value_counts()

non_bowman_years_after = df1[(df1["Sport"] ==
     "Swimming") & (df1["Year"] >=2020) &
     (df1["NOC"] == "USA")]


counts_after =
     non_bowman_years_after["Medal"].value_counts()


data = [
    [150, 49], # Gold
    [73, 57],  # Silver
    [49, 22],  # Bronze
    [116, 88], # No Medal
]

chi2, p_value, dof, expected =
     chi2_contingency(data)

# Print the results
print(f"Chi-square statistic: {chi2}")
print(f"P-value: {p_value}")
print(f"Degrees of freedom: {dof}")
print("Expected frequencies:")
print(expected)

if p_value < 0.05:
    print("Reject  H : There is a statistically
         significant difference.")
else:
    print("Fail to reject H: No statistically
         significant difference.")
```

Listing 2: Chi-Square Test Code

7 References and Appendices

References

[1] Olympics.com. (n.d). The history of the Olympic Games. Retrieved from https://www.olympics.com

[2] Wikipedia. (n.d.). Women in the Olympics. Retrieved from https://en.wikipedia.org/wiki/WomenintheOlympics

[3] History.com. (n.d.). The Olympic Games During the Cold War. Retrieved from https://www.history.com

[4] History.com. (n.d.). Emerging Olympic Powerhouses/ Retrieved from https://www.history.com

[5] Sports Tech Journal. (n.d.). Technology Advancements in the Olympics. Retrieved from https://www.sportstechjournal.com

[6] Bayesian Statistics The Fun Way. (n.d.)

[7] Dive into Data Science, (n.d.)

[8] Python for Data Analysis. (n.d.)

Report on Use of AI

1. OpenAI ChatGPT (2025 version, ChatGPT-4)
    - **Query1**: *Generate Python code for predicting countries likely to win their first Olympic medal.*
    - **Output**: *Python code was generated to filter Olympic data and calculate probabilities based on event success rates and participation. The code identified countries likely to win their first medal.*
2. OpenAI ChatGPT (2025 version, ChatGPT-4)
    - **Query2**: *Provide LaTeX formatting for a report integrating AI and Python predictions.*
    - **Output**: *An example of structured format of LaTeX document template was generated, including sections for Introduction, Objectives, Methodology, and References.*
3. OpenAI ChatGPT (2025 version, ChatGPT-4)
    - **Query3**: *Include results of Python code into the LaTeX document for a full AI-integrated report.*
    - **Output**: *Detailed results and explanations were added to the LaTeX document, focusing on AI-driven insights for Olympic medal predictions.*
4. OpenAI ChatGPT (2025 version, ChatGPT-4)
    - **Query4**: *How to format references and citations in LaTeX.*
    - **Output**: *Guidelines were provided for using BibTeX and thebibliography environments in LaTeX, along with citation examples.*
5. GitHub CoPilot (n.d.)
    - **Usage**: *Auto-completions for Python code used in data analysis*