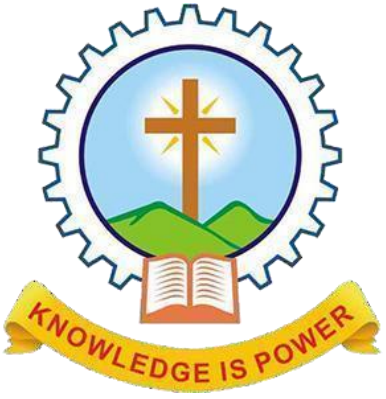
MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM



Department of Computer Applications

Mini Project Report

OLYMPICS DATA ANALYSIS WITH PREDICTION

Done by

VARSHA U

Reg No: MAC22MCA-2028

Under the guidance of
Prof. Manu John

2022-2024

MAR ATHANASIUS COLLEGE OF ENGINEERING
(Affiliated to APJ Abdul Kalam Technological University, TVM)
KOTHAMANGALAM

CERTIFICATE

Olympics Data Analysis with Prediction

Certified that this is the bonafide record of project work done by

Varsha U

Reg No: MAC22MCA-2028

During the academic year 2023-2024, in partial fulfilment of requirements

for award of the degree,

Master of Computer Applications of

APJ Abdul Kalam Technological University

Thiruvananthapuram

Faculty Guide
Prof. Manu John

Head of the Department
Prof. Biju Skaria

Project Coordinator
Prof. Nisha Markose

Internal Examiners

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his divine grace and blessings in making all this possible. May he continue to lead me in the years to come.

I feel enormous pleasure in presenting this report on the “Olympic Data Analysis With prediction”. While presenting this report I benefit from this valuable opportunity to offer my thanks to every one of those who have directed and helped me in getting done with this project done effectively Heading the rundown is respectable Prof. Biju Skaria, Head of the Department, Department of Computer Applications and Prof. Nisha Markose our project coordinator who is a beginner of my motivation.

I would like to thank my project guide, “Prof. Manu John”, for his meaningful idea. Aside from giving me what can be a pleasure for creation, each time he acted expeditiously to address my missteps. The successful fulfilment of this project was achievable by his direction and participation only, without which the work couldn't ever have been finished. At long last, I wish to articulate my profound feeling of respect and appreciation to my parents who hold on to me in any essential condition, and to all others, for saving their time and aiding me to the end of this project in any manner they could.

ABSTRACT

This project involves the development of a Python-based web application designed to analyse Olympic data, with a particular focus on leveraging the Kaggle 120 years of Olympic history dataset. The application empowers users to interactively explore various data points, including countries, sports, events, and medal counts, through dynamic visualizations such as graphs, tables, and maps. Utilizing Python libraries like Pandas, Matplotlib, and seaborn, the application ensures robust data analysis and visualization capabilities.

In addition to offering an intuitive user interface for exploring Olympic trends and statistics, the project extends its functionality to include a novel aspect—medal prediction. Leveraging advanced machine learning techniques, specifically XGBoost and Random Forest algorithms, the application forecasts the likelihood of athletes winning medals in future Olympic Games. A meticulous comparison process is employed to select the most effective model, providing users with insightful predictions on athlete success.

This project not only showcases the versatility of Python in creating interactive data analysis applications but also highlights its potential across diverse domains, ranging from sports analysis to business intelligence and scientific research. Through seamless integration of data exploration and predictive modelling the web application demonstrates the power of Python in unlocking valuable insights from Olympic datasets.

LIST OF TABLES

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

The Olympics stands as a paramount global event, serving as a universal platform where athletes from diverse nations showcase their talents. This grand sporting spectacle, held biennially with both summer and winter games, represents one of the largest and most prestigious gatherings in the world. Analysing the wealth of Olympic data provides invaluable insights into athlete performance, historical trends, and factors influencing game outcomes. Exploratory data analysis proves pivotal in unravelling statistical and visual representations, offering a profound understanding of the Olympics' evolution.

As we delve into the evolution of Olympic Games, numerous scenarios come to light, encompassing changes in participant numbers, event dynamics, expenditure costs, performance enhancements, and the impact of external factors like pandemics. These analyses not only illuminate the past and present but also offer a predictive lens into the future of the Olympics.

In this endeavour, we gathered datasets from Kaggle, focusing on the top ten sports in the United States, as determined by their average viewership ranking. This strategic selection aims to provide a comprehensive analysis of past Olympic performances in widely followed sports. Adding a distinctive layer to this exploration, we introduce a winning chance prediction component to our project. Employing advanced machine learning models, such as XGBoost and Random Forest, we aim to forecast the likelihood of athletes winning medals in future Olympic Games. By seamlessly integrating this predictive element, our analysis gains depth, offering insights into both historical achievements and future predictions. This dual focus on exploration and prediction positions our project at the forefront of Olympic data analysis, promising valuable insights for athletes, nations, and enthusiasts alike.

2. SUPPORTING LITERATURE

2.1 Literature Review

Paper 1: Sagala, Noviyanti TM, and Muhammad Amien Ibrahim. "A Comparative Study of Different Boosting Algorithms for Predicting Olympic Medal." ***2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*****. IEEE, 2022**

The primary objective of this paper was to evaluate the performance of three prominent boosting algorithms, namely LGBM, XGBoost, and CatBoost, in the task of predicting whether an athlete had a likelihood of winning a medal in the Olympic Games. To accomplish this, the researchers leveraged an extensive dataset that spanned 11 Olympic Games, encompassing a wide range of athlete attributes such as nationality, sport, age, and historical performance. The methodology employed in this study involved an initial round of training using default algorithm parameters, followed by a meticulous hyperparameter tuning process through Grid Search.

Upon analysing the results of their experiment, it became evident that XGBoost stood out as the leading performer among the three boosting algorithms. After the hyperparameter tuning phase, XGBoost yielded remarkable results, achieving an accuracy rate of 90%, a precision score of 96%, a recall rate of 83%. These outcomes underscore the remarkable effectiveness of boosting algorithms when it comes to the precise prediction of Olympic medal outcomes. In conclusion, this study firmly establishes XGBoost as the preeminent model for the task, highlighting its potential for enhancing the accuracy of Olympic medal predictions, a significant feat in the realm of sports.

Paper 2: - Jia, Mengjie, et al. Jia, Mengjie, et al. "A Random Forest Regression Model Predicting the Winners of Summer Olympic Events." ***Proceedings of the 2020 2nd International Conference on Big Data Engineering*****. 2020.**

The study led by Mengjie Jia, Yue Zhao, Furong Chang, Bofeng Zhang from Shanghai University, and Kenji Yoshigoe from Toyo University, Tokyo, is centered on the prediction of Summer Olympic event winners, employing a Random Forest Regression model. Their research draws upon a comprehensive dataset spanning multiple Summer Olympic Games which contains information and awards of all athletes participating in Olympic Games from 1896 to 2016 and also incorporates a diverse set of factors such as GDP, population, national team size, and the influence of home advantage as key parameters to construct their predictive model. Moreover, the researchers utilize the FP-growth algorithm to discern association rules within the dataset, adding an insightful layer to their analysis.

One of the notable achievements of this research is its ability to not only investigate the impact of economic factors on Olympic event outcomes but also provide precise predictions for winners in specific events during the 2020 Tokyo Olympic Games. Impressively, their model achieves an accuracy rate of 89.76%, underscoring its effectiveness in forecasting the results of Olympic events. This research not only contributes valuable insights into the multifaceted factors influencing Olympic success but also demonstrates the practical application of machine learning in the realm of sports analytics, emphasizing the potential for data-driven predictions to enhance our understanding of sporting events and outcomes.

Paper 3. Schlembach, Christoph, et al. "Forecasting the Olympic medal distribution–a socioeconomic machine learning model." *Technological Forecasting and Social Change* 175 (2022): 121314.

The research paper titled "Forecasting the Olympic medal distribution – A socioeconomic machine learning model" is dedicated to the development and presentation of a robust machine learning approach designed for predicting the distribution of Olympic medals. To accomplish this ambitious goal, the authors employ a two-staged Random Forest algorithm, meticulously trained on an extensive dataset encompassing socioeconomic attributes of 206 countries spanning the years from 1991 to 2020. This comprehensive methodology entails not only predicting the total medal count per country but also the distribution of medals across different sports. Their dataset incorporates a wide array of variables, including population size, GDP per capita, and investments in sports infrastructure, sourced from reputable organizations such as the World Bank, International Olympic Committee, and Organisation for Economic Co-operation and Development. One of the standout achievements of this study is the demonstrated accuracy of their machine learning model through rigorous comparisons with actual medal counts from multiple Olympic Games, including those held in 2008, 2012, 2016, and 2020. The machine learning model consistently outperforms conventional forecasting methods, boasting an impressive accuracy rate of 85.38%. Furthermore, the research highlights the model's capability in identifying crucial factors linked to Olympic success, such as population size, GDP per capita, and investments in sports infrastructure. This underscores the immense potential of machine learning in providing precise predictions for Olympic outcomes and positions it as a valuable tool for various stakeholders involved in the world of sports and policymaking.

Summary Table

	TITLE	YEAR	PUBLISHER	SUMMARY
PAPER 1	**A Comparative Study of Different Boosting Algorithms for Predicting Olympic Medal**	**2022**	**IEEE**	**Algorithm: XGBoost** **Accuracy: 90%** **Dataset: Records of Olympics history from the earliest competition in 1896 to recent games in 2016.**
PAPER 2	**A Random Forest Regression Model Predicting the Winners of Summer Olympic Events.**	**2020**	**ACM (Association for Computing Machinery)**	**Algorithm: random forest** **Accuracy: 89.76** **Dataset: information on athletes participating in the 1896 to 2016 Winter and Summer Olympic Games.**
PAPER 3	**Forecasting the Olympic medal distribution–a socioeconomic machine learning model**	**2022**	**Elsevier**	**Algorithm: Random Forest model** **accuracy: 85.38%.** **Dataset: GDP, population, the number of athletes, the impact of COVID-19, host country status, political regime, and geographic region.**

Table 2.1: Summary of Literature Review

2.2 Findings and Proposals

Upon reviewing the three papers on Olympic medal prediction, it is apparent that diverse methodologies and algorithms have been explored. While each paper presents unique approaches, a common thread is the application of machine learning, particularly random forest (RF) regression models and XGBoost, to forecast Olympic medal outcomes. The first paper introduces a comprehensive socioeconomic model, the second explores boosting algorithms (including XGBoost), and the third incorporates a two-staged random forest algorithm. Despite the diversity in approaches, the third paper's use of a two-staged random forest algorithm demonstrated superior accuracy.

However, it's essential to weigh the advantages and limitations of both XGBoost and Random Forest. The first paper doesn't explicitly use these algorithms, but meta-learning algorithms are suggested. The second paper employs XGBoost and other boosting algorithms, highlighting their advantages but acknowledging the time-consuming nature of text filtering. The third paper uses Random Forest and acknowledges its time-consuming aspect as a limitation.

In the comparison, both XGBoost and Random Forest have demonstrated effectiveness in predictive modeling. However, considering the observed advantages and limitations, the project leans towards a comprehensive comparison between XGBoost and Random Forest for Olympic medal prediction. The selection between the two will be based on factors such as accuracy, efficiency, and suitability for handling the complexity of the dataset. Future work may involve fine-tuning hyperparameters and exploring ensemble methods to enhance predictive performance. This iterative approach aligns with the goal of selecting the algorithm that provides the optimal balance between accuracy and computational efficiency for the specific task of Olympic medal prediction.

3. SYSTEM ANALYSIS

3.1 Analysis of Dataset

I collect my datasets from Kaggle.
https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results

This project will investigate the enormous history of Olympic Games to see if it can predict whether an athlete will win a medal. The history of the Olympics is influenced by several variables. Before creating a predictive model, we need to undertake a thorough data examination to determine these factors. For this study, we used a previous Olympics history dataset. The dataset is collected from Kaggle. It contains two files: athlete_events.csv and noc_regions.csv. The dataset contains details about all the Olympic Games from Athens 1896 to Rio 2016 which includes information about the athletes, the events, the medals, and the countries that have participated in the Olympics.

3.1.1 About the Dataset

athlete_events.csv:

The file athlete_events.csv contains 271116 rows and 15 columns. Each row corresponds to an individual athlete competing in an individual Olympic event. (athlete-events)

The columns are:

- ID - Unique number for each athlete
- Name - Athlete's name

- Sex - M or F
- Age - Integer
- Height - In centimetre’s
- Weight - In kilograms
- Team - Team name
- NOC - National Olympic Committee 3-letter code
- Games - Year and season
- Year - Integer
- Season - Summer or Winter
- City - Host city
- Sport - Sport
- Event - Event
- Medal - Gold, Silver, Bronze, or NA

noc_regions.csv:

The file noc_regions.csv contains 230 rows and 3 columns. Each row corresponds to an individual region.

The columns are:

- NOC (National Olympic Committee 3 letter code)
- Country name (matches with regions in map_data("world"))
- Notes

Figure 3.1: Snapshot of merging datasets

1.1.2 Explore the Dataset

The expected type of values for each feature is as follows:

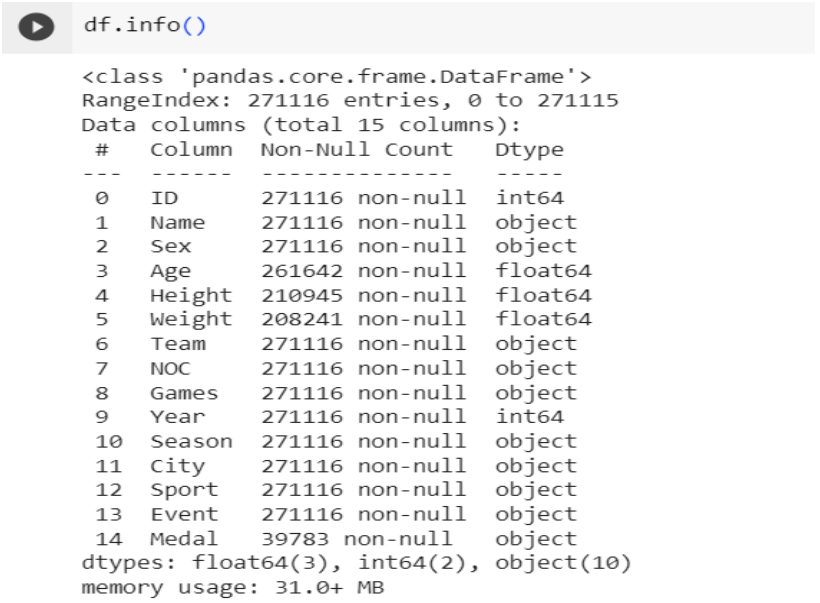


Figure 3.2 : Data frame information

An examination was conducted to look at the distribution of float data in the dataset such as athletes' age, height, and weight. Fig. 3.1.1 shows that the average age of the Olympic Athletes is 25 years and 7 months old with a median value of 24 years, making the distribution right skewed. Fig. 3.1.1 also shows the distribution of the height of the athletes which is around 175.37 centimeters with a slightly similar median value at 173

centimeters, which rather resembles a normal distribution. Similarly, the distribution of athlete's weight was the average value at 70.69 kilograms while the median is 70 kilograms.

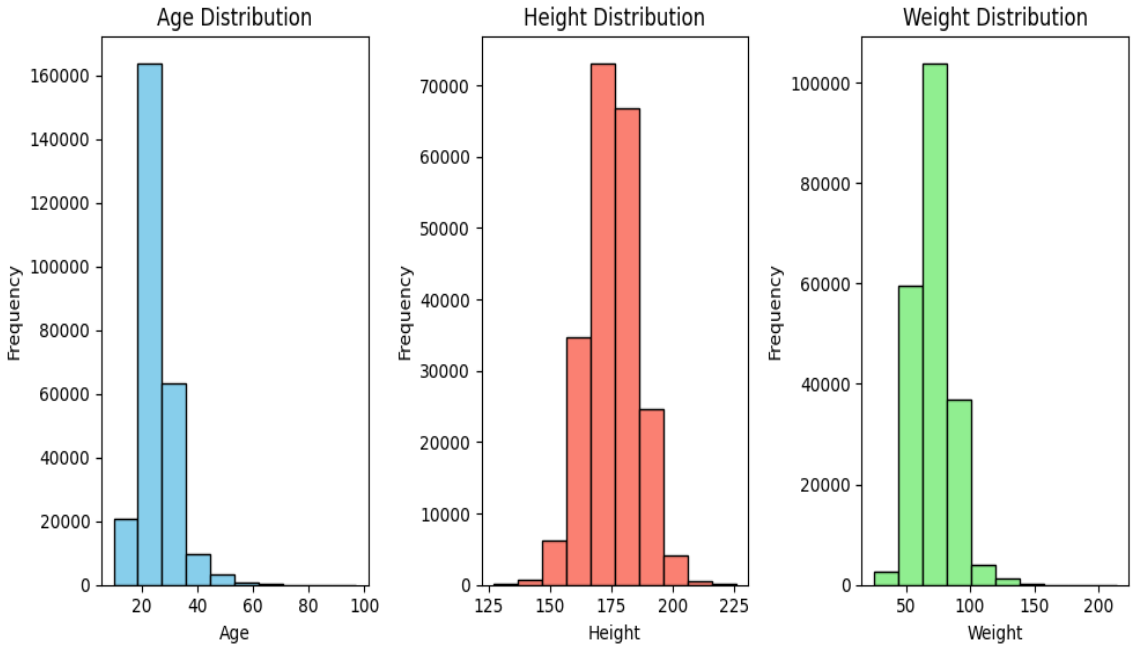


Figure 3.3: distribution of float data in the dataset

Despite the histograms can provide insights regarding the physicality aspects of the athletes, these three float variables are the only numerical variables with missing values. There are about 3.5%, 22%, and 33% missing values for each of age, height, and weight variables respectively. In this study, the missing values are then imputed with the mean value taken from each of these three float variables. Furthermore, the medal variable also contains missing values. It is important to note that the medal variable only keep records of athletes having won a medal. As a result, an empty record will be given to those atletes that didn't win the competition. In this case, the missing values in the medal variable is imputed with a label of "None".

There are several variables that indicate location such as team, NOC, and city. The team variable represents the country or organization that they represent. Previously, there are certain cases where people that live in minor islands also participated in the Olympics, thus they represent an organization rather than a country. Currently, most of these minor islands have become part of countries known nowadays. However, as a categorical variable, the team variable is considered to contain a high variance as there are more than 600 different teams.

In terms of sport, there are 56 different sports the Olympics. This categorical variable is having to many variations.

The city variable represents the city where the Olympics was held geographically. However, there are as many as 42 cities in the dataset.

In this study, several variables have been selected for modelling. The independent variables are age, height, weight, sex, city, sport, and region. Other variables such an event is not included since it represents a deeper classification of sport. All the independent variables are then one-hot-encoded, increasing the size of the dataset horizontally. ln contrast, the medal variable is selected as the dependent variable. There are four attributes in medal variable such as gold, silver, bronze, none.

There are several variables that indicate location such as team, NOC, and city. The team variable represents the country or organization that they represent. Previously, there are certain cases where people that live in minor islands also participated in the Olympics, thus they represent an organization rather than a country. Currently, most of these minor islands have become part of countries known nowadays. However, as a categorical variable, the team variable is considered to contain a high variance as there are more than 600 different teams.

In terms of sport, there are 56 different sports the Olympics. This categorical variable is having to many variations.

The city variable represents the city where the Olympics was held geographically. However, there are as many as 42 cities in the dataset.

In this study, several variables have been selected for modelling. The independent variables are age, height, weight, sex, city, sport, and region. Other variables such an event is not included since it represents a deeper classification of sport. All the independent variables are then one-hot-encoded, increasing the size of the dataset horizontally. ln contrast, the medal variable is selected as the dependent variable. There are four attributes in medal variable such as gold, silver, bronze, none.

```
Number of Unique Values in Each Column:
ID        135571
Name      134732
Sex            2
Age           74
Height        95
Weight       220
Team        1184
NOC          230
Games         51
Year          35
Season         2
City          42
Sport         66
Event        765
Medal          3
region       205
dtype: int64
```

Figure 3.4 : Counts of unique values

3.2 Data Preprocessing

3.2.1. Data Cleaning

Data Cleaning is the data pre-processing method we choose. Data cleaning routines attempt to fill in missing values, smooth out noisy data, and correct inconsistencies. The dataset taken is already pre-processed, so pre-processing techniques are not need for the dataset. But for assurance pre-processing techniques for handling missing values and duplicated values are made.

Outliers:

To identify outliers, you can perform statistical analysis or visualization techniques such as box plots or scatter plots for each numerical attribute (e.g., age, height, weight). Outliers are data points that significantly deviate from the typical range. Handling outliers can depend on the specific analysis or modelling task.

Missing Values:

To check for missing values, examine each column for any NaN or null values. If you find any, you may need to decide on an appropriate strategy for handling missing data, such as imputation or removing rows with missing values.

Consistency:

Data consistency can be ensured by examining categorical variables (e.g., sex, region, sport, city, medal) for consistent coding and numerical variables (e.g., age, height ,weight) for reasonable ranges and units. Consistency checks are essential to ensure that the data is reliable and accurate for analysis.

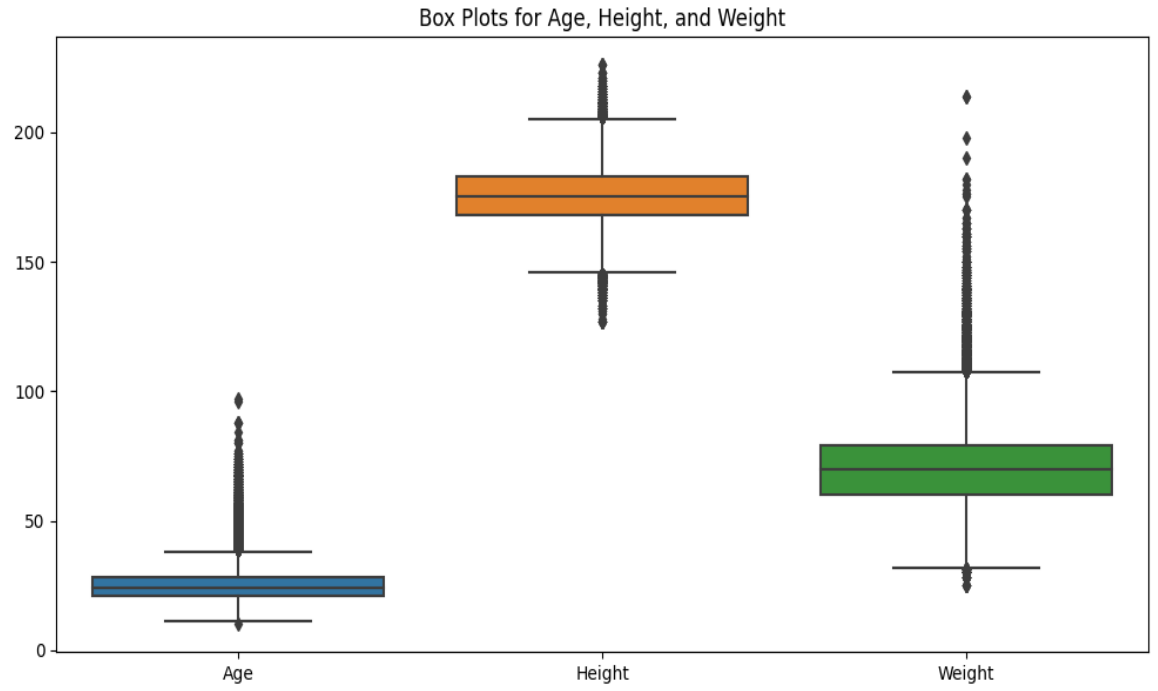


Figure 3.5: Box plot for outliers in dataset

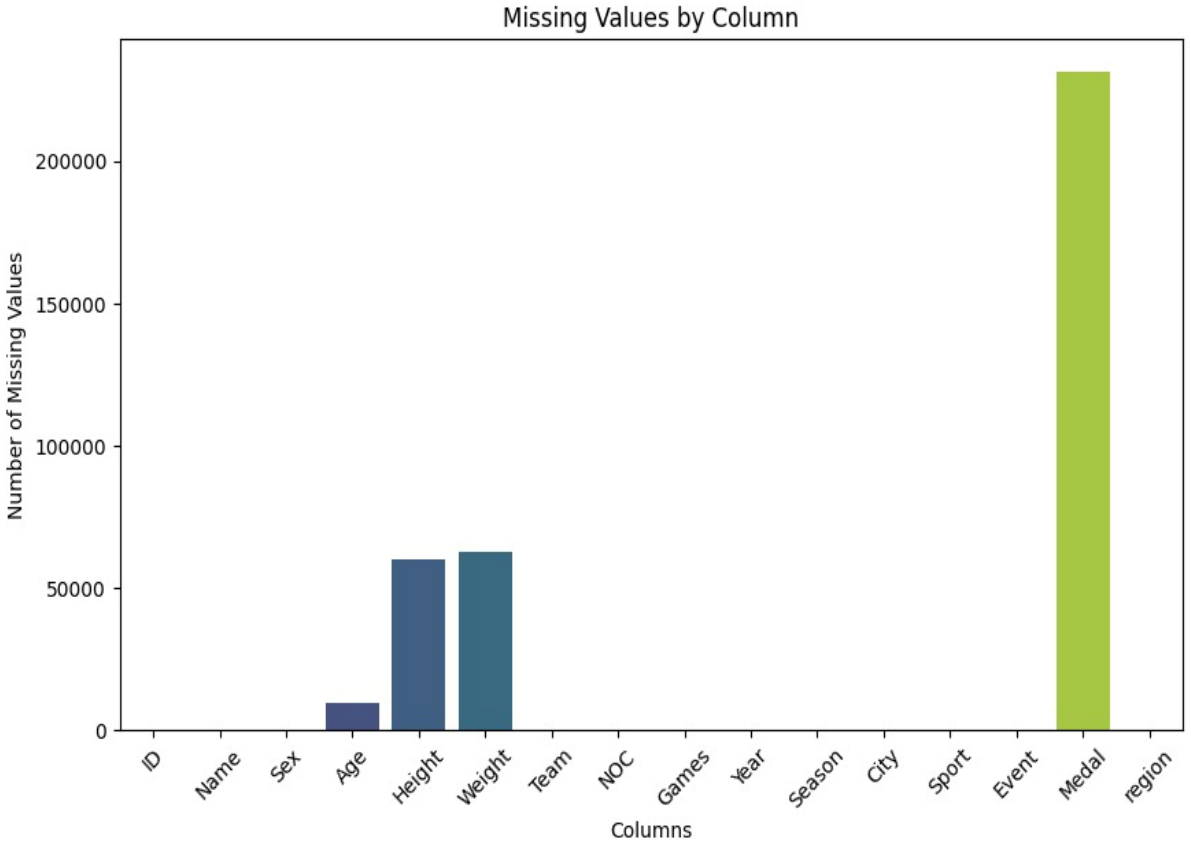


Figure 3.6: Missing values in dataset

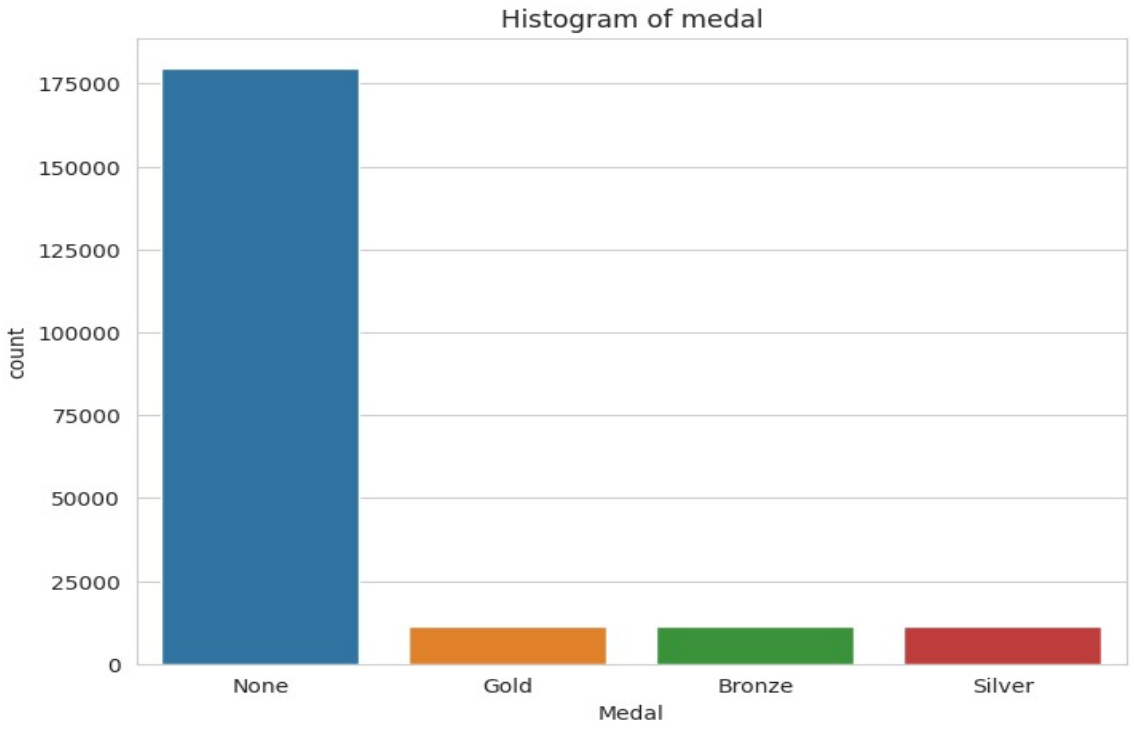


Figure 3.7: Inconsistency of medal column

The preprocessing of the Olympic dataset involves a systematic approach to enhance its quality and reliability. Initially, a thorough inspection for missing values was conducted, revealing that certain records in the dataset lacked essential information. To ensure a comprehensive analysis, the focus was narrowed down to the 'Summer' season, and records with missing values in crucial columns such as 'Age,' 'Height,' 'Weight,' and 'Medal' were selectively dropped. This meticulous step aimed to create a more robust dataset for subsequent analyses.

Outliers, often potential sources of distortion in data, were diligently addressed using the Interquartile Range (IQR) method. The function 'remove_outliers_iqr' was implemented to systematically identify and exclude data points lying outside the IQR boundaries for numeric columns, namely 'Age,' 'Height,' and 'Weight.' This approach effectively mitigates the impact of extreme values on the overall dataset, ensuring a more accurate representation of the underlying patterns.

To further refine the dataset, duplicate records were identified and subsequently removed. This step safeguards against redundancy and maintains the integrity of the

data. In tandem with these cleaning efforts, standardization of 'City' names was carried out, with 'Torino' being replaced by 'Turin' to establish uniformity and clarity within the dataset.

A keen examination of the distribution of 'Medal' values provided valuable insights into the prevalence of medals across different events and categories, setting the stage for more nuanced analyses.

The subsequent focus turned towards handling missing values in both numeric and categorical columns. Leveraging the SimpleImputer, missing numeric values were imputed with the mean, ensuring a data-driven approach to fill gaps. Simultaneously, categorical columns were handled using a constant imputation strategy, assigning the value 'None' to missing entries. This strategic handling of missing values contributes to a more complete and coherent dataset.

As a final step, the dataset underwent a thorough conversion of numeric columns to the integer data type. This conversion aligns with the nature of the data, facilitating a more efficient representation for subsequent analyses.

The arrangement of columns was meticulously considered to optimize the dataset's organization and readability. The resulting cleaned dataset, named 'clean_athlete.csv,' serves as a foundation for insightful analyses, free from the uncertainties and inconsistencies that may arise from missing or distorted data.

To rectify class imbalance in the 'Medal' column, the Random OverSampler from imbalanced-learn was applied, allowing for the oversampling of minority classes, and thereby creating a more balanced dataset. By separating features ('A') and the target variable ('B'), oversampling was conducted, resulting in resampled features ('A_resampled') and target variable ('B_resampled'). The concatenation of these resampled components formed a new balanced DataFrame ('data1'), ensuring a more equitable representation of different medal classes. The subsequent display of value counts in the target variable validated the efficacy of the oversampling process, affirming the successful mitigation of class imbalance.

The iterative and comprehensive nature of these preprocessing steps underscores their critical role in ensuring the dataset's reliability and fitness for subsequent analytical endeavors.

Selecting only summer Olympics:

```
df=df[df['Season']=='Summer']
```

Removing outliers and duplicates:

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

# Select the numeric columns
numeric_columns = df[['Age', 'Height', 'Weight']]

# Define a function to remove outliers using the IQR method
def remove_outliers_iqr(data, column_name):
    Q1 = data[column_name].quantile(0.25)
    Q3 = data[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[column_name] >= lower_bound) & (data[column_name] <= upper_bound)]

# Remove outliers for each column
numeric_columns_cleaned = numeric_columns.copy()
for column_name in numeric_columns.columns:
    numeric_columns_cleaned = remove_outliers_iqr(numeric_columns_cleaned, column_name)
```

```
df.drop_duplicates(keep='first', inplace=True)
```

```
# get the categorical and numeric column names
obj_col = list(df.select_dtypes(include='object').columns)
num_col = list(df.select_dtypes(exclude=['object']).columns)
print(f'The object columns are: {obj_col} \nThe numeric columns are: {num_col}')
```

```
The object columns are: ['Name', 'Sex', 'Team', 'NOC', 'Games', 'Season', 'City', 'Sport', 'Event', 'Medal', 'region']
The numeric columns are: ['ID', 'Age', 'Height', 'Weight', 'Year']
```

```
# Get the dataframe of the numeric and objects
df_obj = df.select_dtypes(include='object')
df_num = df.select_dtypes(exclude='object')
```

Handling Missing values:

```
# Simple Imputer for the Numeric
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
imp_mean.fit(df_num)
#imp_mean.get_params()

# Prepare the numerica Data
nums = pd.DataFrame(data = imp_mean.transform(df_num), columns=num_col)

# Simple Imputer for the category
imp_miss = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value='None')
imp_miss.fit(df_obj)

# prepare the category data
obj = pd.DataFrame(data = imp_miss.transform(df_obj), columns=obj_col)

data = nums.join(obj)

# Converting numeric columns to Integer
data['ID'] = data['ID'].astype('int64')
data['Age'] = data['Age'].astype('int64')
data['Height'] = data['Height'].astype('int64')
data['Weight'] = data['Weight'].astype('int64')
data['Year'] = data['Year'].astype('int64')
```

Making data consistent:

```
value_counts = data['Medal'].value_counts()
print(value_counts)
```

```
None      179741
Gold       11456
Bronze     11409
Silver     11212
Name: Medal, dtype: int64
```

```
from imblearn.over_sampling import RandomOverSampler
import pandas as pd

# Assuming 'Medal' is the column indicating the medal (0 for bronze, 1 for gold, 2 for none, 3 for silver)

# Separate features and target variable
A = data.drop('Medal', axis=1)
B = data['Medal']# Use RandomOverSampler to balance the classes
ros = RandomOverSampler(random_state=42)
A_resampled, B_resampled = ros.fit_resample(A, B)

# Create a new balanced DataFrame
data1 = pd.concat([A_resampled, B_resampled], axis=1)

# Display the value counts of the target variable in the balanced DataFrame
print(data1['Medal'].value_counts())
```

```
None      179741
Gold      179741
Bronze    179741
Silver    179741
Name: Medal, dtype: int64
```

```
# Arranging the columns
data = data[['ID', 'Name', 'Age', 'Sex', 'Height', 'Weight', 'Year', 'Team', 'NOC', 'region',
       'Games', 'Season', 'City', 'Sport', 'Event', 'Medal']]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 213818 entries, 0 to 213817
Data columns (total 16 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      213818 non-null  int64
 1   Name    213818 non-null  object
 2   Age     213818 non-null  int64
 3   Sex     213818 non-null  object
 4   Height  213818 non-null  int64
 5   Weight  213818 non-null  int64
 6   Year    213818 non-null  int64
 7   Team    213818 non-null  object
 8   NOC     213818 non-null  object
 9   region  213818 non-null  object
 10  Games   213818 non-null  object
 11  Season  213818 non-null  object
 12  City    213818 non-null  object
 13  Sport   213818 non-null  object
 14  Event   213818 non-null  object
 15  Medal   213818 non-null  object
dtypes: int64(5), object(11)
memory usage: 26.1+ MB
```

```
data.to_csv('/content/clean_athlete.csv', index=False)
```

Figure 3.8: Data preprocessing

3.2.2 Analysis of Feature Variables

A feature is a measurable property of the object you're trying to analyze. In datasets, features appear as columns. Features are the basic building blocks of datasets. The quality of the features in your dataset has a major impact on the quality of the insights you will gain when you use that dataset for machine learning. Additionally, different business problems within the same industry do not necessarily require the same features, which is why it is important to have a strong understanding of the business goals of your data science project.

Upon inspecting the information about feature variables, it was revealed that the dataset contains a mix of numeric and categorical features, reflecting various athlete attributes and contextual information associated with Olympic events. This diverse set of features provides a comprehensive basis for predicting medal outcomes, encompassing aspects such as age, height, weight, team affiliation, and event specifics.

A glimpse into the first five rows of the feature variables reveals a snapshot of the dataset, showcasing the initial instances and their corresponding feature values. This preliminary exploration serves as a foundation for the subsequent analysis of feature variables, offering insights into the types of data that will be utilized for model training and prediction in the context of Olympic medal outcomes.

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 718964 entries, 0 to 718963
Data columns (total 7 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Age     718964 non-null  int64
 1   Sex     718964 non-null  object
 2   Height  718964 non-null  int64
 3   Weight  718964 non-null  int64
 4   region  718964 non-null  object
 5   City    718964 non-null  object
 6   Sport   718964 non-null  object
dtypes: int64(3), object(4)
memory usage: 38.4+ MB
```

```
X.head()
```

	Age	Sex	Height	Weight	region	City	Sport
0	24	M	180	80	China	Barcelona	Basketball
1	23	M	170	60	China	London	Judo
2	24	M	175	70	Denmark	Antwerpen	Football
3	34	M	175	70	Denmark	Paris	Tug-Of-War
4	18	F	168	70	Netherlands	Los Angeles	Athletics

Figure 3.9: Feature list

3.2.3 Analysis of Class Variables

The 'Medal' column, representing the target variable, has been encoded into integer values using Label Encoding. This transformation assigns numerical labels to each class within the 'Medal' column, facilitating the compatibility of the data with machine learning models that require numeric inputs. However, an intriguing aspect of the initial class distribution emerges upon closer inspection of the encoded 'Medal' values. The label 'None,' indicative of instances where athletes did not win a medal, is represented by the value '0.' Conversely, 'Gold' is denoted by '1,' and other medal classes such as 'Silver' and 'Bronze' are also assigned numerical labels. This transformation allows for a more seamless integration of the target variable into machine learning algorithms, paving the way for accurate model training and prediction based on the historical Olympic dataset.

```
# Encode the 'Medal' column to integers using Label Encoding
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

```
y.head()
```

```
0    None
1    None
2    None
3    Gold
4    None
Name: Medal, dtype: object
```

Figure 3.10: class label

3.3 Data Visualization

Data visualization is the graphical representation of information and data in a pictorial or graphical format (Example: charts, graphs, and maps). Data visualization tools provide an accessible way to see and understand trends, patterns in data, and outliers. Data visualization tools and technologies are essential to analysing massive amounts of information and making data driven decisions. The concept of using pictures is to understand data that has been used for centuries. General types of data visualization are Charts, Tables, Graphs, Maps, and Dashboards.

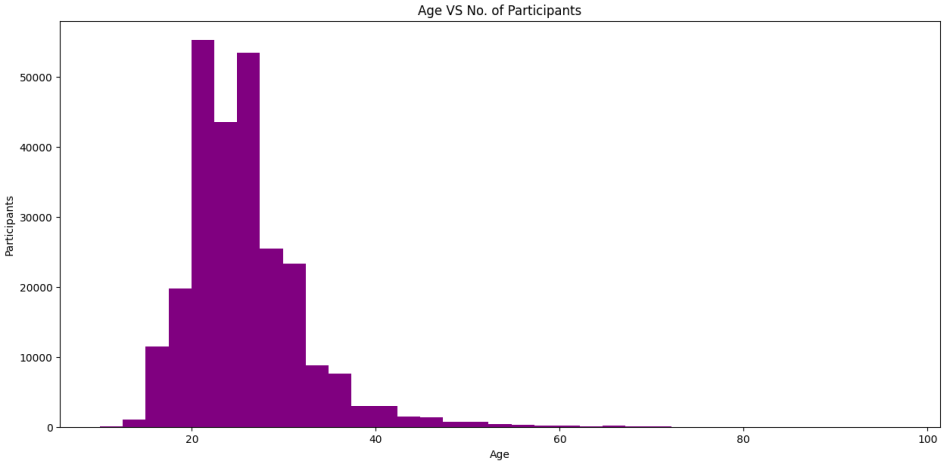


Figure 3.11: Age distribution

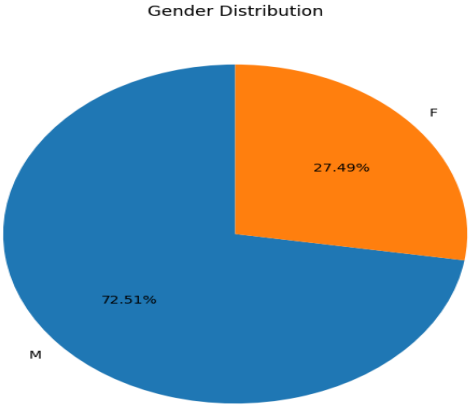


Figure 3.12: pie chart showing gender discrimination.

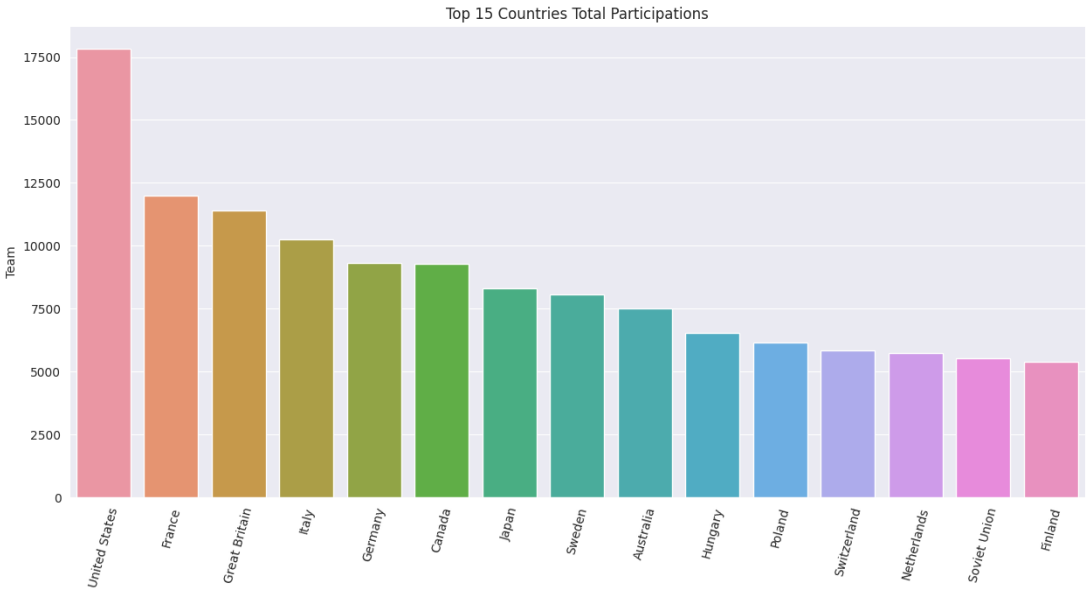


Figure 3.13: Histogram showing total participants in top 15 countries

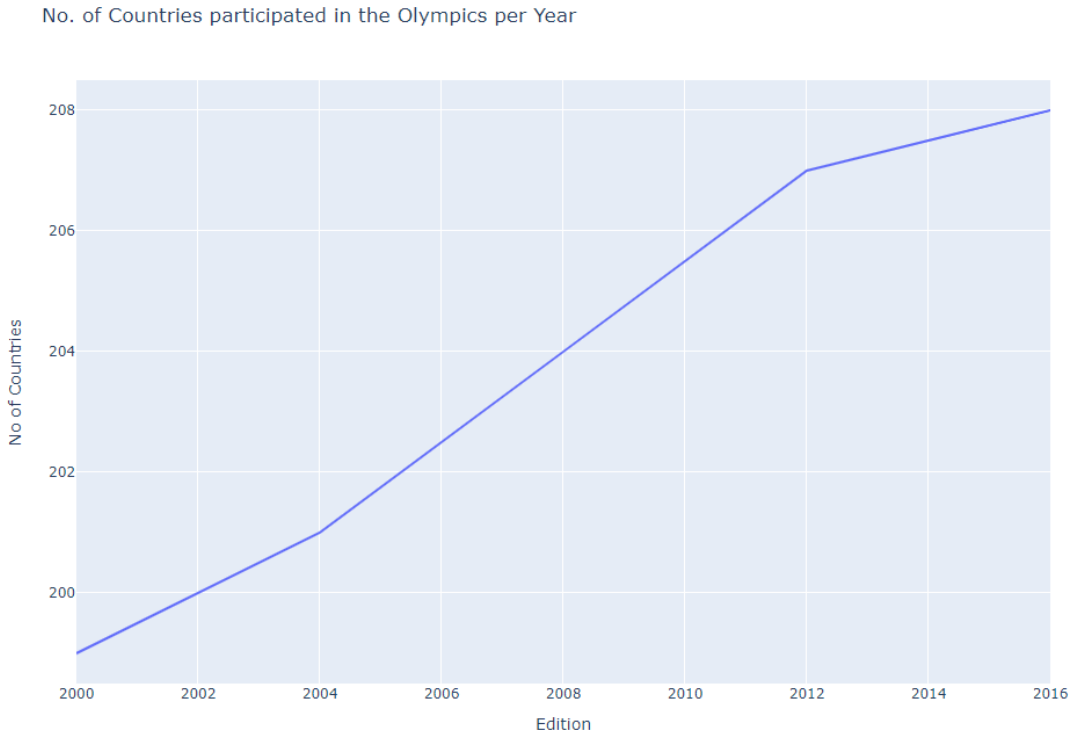


Figure 3.14: Line graph showing number of countries per year

3.4. Analysis of Algorithms

Proposed Algorithms:

1) Random Forest:
2) XGBoost

We employed Random Forest and XGBoost algorithms for a predictive task, comparing their accuracies to determine the most effective model. The algorithm demonstrating higher accuracy will be chosen for making predictions, ensuring superior performance in our application. This approach allows us to select the most suitable algorithm for optimizing predictive outcomes.

3.4.1 Random Forest:

Random Forest is a powerful and versatile ensemble learning algorithm used for both classification and regression tasks in machine learning. It's an extension of the decision tree algorithm and is known for its robustness and ability to handle complex data and high-dimensional feature spaces. Overall, Random Forest is a versatile and widely used algorithm known for its accuracy, robustness, and applicability to a wide range of machine learning tasks. It is particularly valuable when dealing with complex and highdimensional datasets. Random Forest's combination of randomization, ensemble learning, and decision trees makes it a robust and widely used algorithm in machine learning, suitable for both beginners and experienced data scientists.

Features:

- **Ensemble Learning:** Random Forest is an ensemble learning method that combines the predictions of multiple decision trees to improve overall predictive accuracy and reduce overfitting.
- **Decision Trees as Base Learners:** The base learners (individual models) in Random Forest are decision trees. Each decision tree is trained independently on a bootstrapped subset of the training data.

- **Random Sampling (Bootstrapping):** During training, Random Forest randomly selects a subset of the training data (with replacement) for each decision tree. This random sampling ensures diversity in the training datasets used for individual trees.
- **Random Feature Selection:** At each node of a decision tree, only a random subset of features (variables) is considered for splitting. This reduces the correlation between trees and prevents dominance by a single feature.
- **Aggregation of Predictions:** For classification tasks, Random Forest combines the predictions of individual trees by taking a majority vote (mode) to make the final prediction. For regression tasks, it averages the predictions of individual trees to obtain the final regression output.
- **Robustness and Generalization:** Random Forest is robust to noise and outliers in the data. It tends to generalize well and is less prone to overfitting compared to a single decision tree.
- **High Predictive Accuracy:** Random Forest often achieves high predictive accuracy and is considered a strong performer in many machine learning competitions and real-world applications.
- **Feature Importance:** Random Forest provides a measure of feature importance. You can analyze the importance scores to understand which features contribute most to the model's predictions.
- **Out-of-Bag (OOB) Error Estimation:** Random Forest uses the OOB error estimate, which is the error rate calculated on the data points that were not included in the bootstrapped samples. This provides an internal estimate of the model's performance.
- **Parallel Processing:** Random Forest can be easily parallelized because the individual decision trees are trained independently. This makes it computationally efficient and suitable for large datasets.
- **Handling Missing Values and Outliers:** Random Forest can handle missing values and outliers gracefully, as it uses ensemble methods to make predictions. It does not rely on a single tree's behaviour.

- **Hyperparameter Tuning:** Random Forest has hyperparameters that can be tuned for optimal performance, such as the number of trees, the maximum depth of trees, and the number of features considered at each split.
- **Interpretability:** While individual decision trees are interpretable, the interpretation of a Random Forest model can be more challenging due to the ensemble nature. However, feature importance scores can provide insights into variable importance.
- **Versatility:** Random Forest is versatile and can be applied to various machine learning tasks, including classification, regression, and feature selection.

3.4.1.1 Activity Diagram

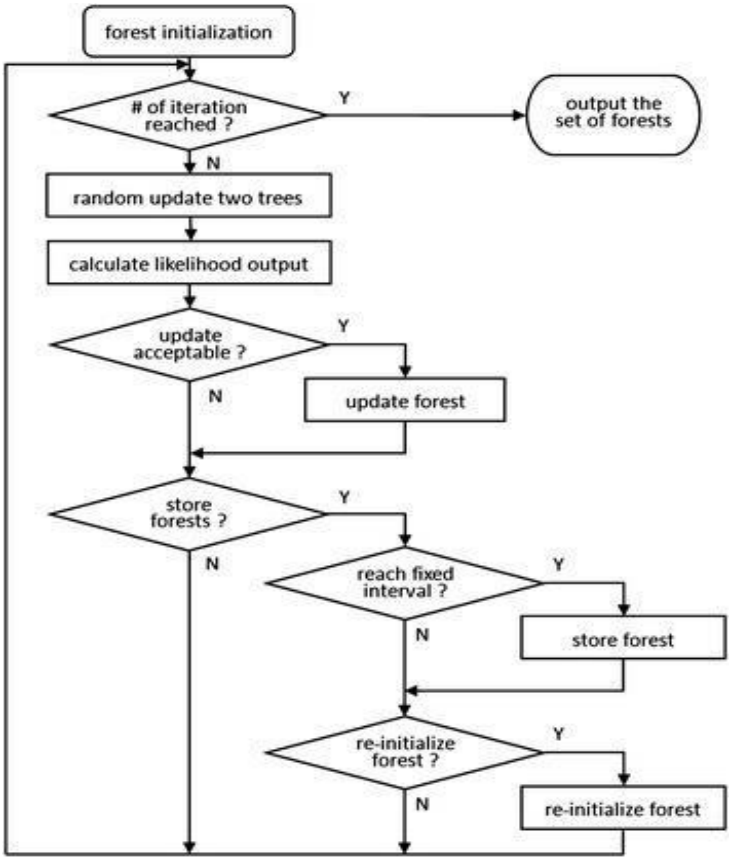


Figure 3.15: Random Forest Activity Diagram

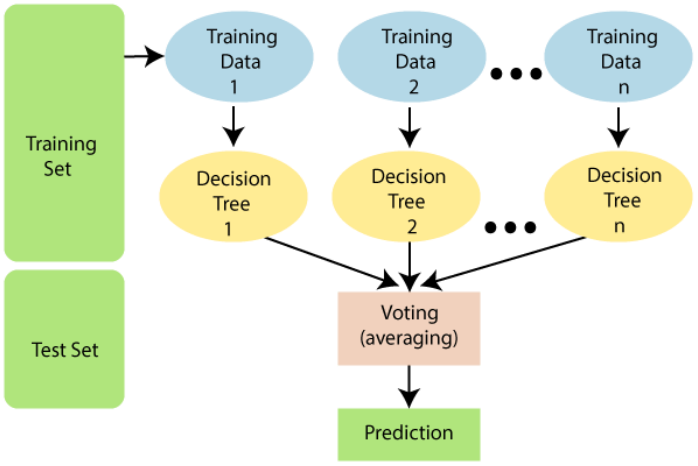


Figure 3.16: Diagrammatic Representation of Random Forest

Working Process of Random Forest:

- **Step-1:** Select random K data points from the training set.
- **Step-2:** Build the decision trees associated with the selected data points (Subsets).
- **Step-3:** Choose the number N for decision trees that you want to build.
- **Step-4:** Repeat Step 1 & 2.
- **Step-5:** For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

3.4.2 XGBoost:

XGBoost denotes extreme Gradient Boosting. XGBoost is implementation of gradient boosting algorithms. It is available in many forms like tool, library etc. It mainly focuses on model performance and computational time. It greatly reduces the time and greatly lifts the performance of the model. Its implementation has the features of scikit-learn and R implementations and have a newly added features like regularization. Regularized gradient boosting means gradient boosting with both L1 and L2 type regularizations. The main best features that the implementation of the algorithm provides are: Automatic handling of missing values with sparse aware implementation and it provides block structure to promote parallel construction of

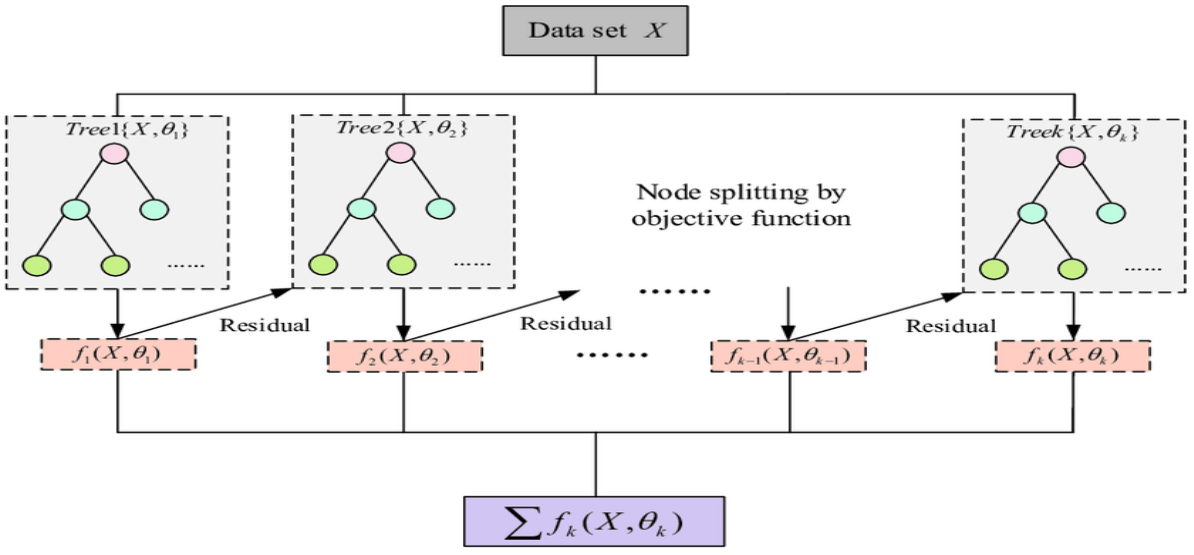
tree and continued training which supports further boost an already fitted model on fresh data. Gradient boosting is a technique where new models are made that can predict the errors or remains of previous models and then added together to make and final prediction, they use gradient descendent algorithms to reduce loss during adding of new models. The support both classification and regression type of challenges. In the training part generally an object function is defined.

- XG Boost: It works on gradient boosting algorithm
- Gradient boosting algorithm works on the basic principle gradient descent. This model is built using tree-based learners (Decision Trees)
- XGradient boosting Algorithm:
- Final prediction=Base value (the starting prediction from basic decision tree) +LR*w1+LR*w2+..+LR*wn

  Where LR= learning rate=eta

  w1=residual predicted value by 1st residual model

  wn=residual predicted value by nth residual model
- Xgboost is different from other gradient boost is because of its tuning parameters Intelligent Career Guidance System.
- The main tuning parameters are regularisation parameter (Lambda), threshold that defines auto pruning (Gamma), Learning rate(eta)



Accuracy Comparison

Accuracy is compared after training the model using both Random Forest Algorithm and K-NN algorithm.

Random Forest	91.8%
XGBoost	91.1%

Table 3.1: Accuracy Comparison

```
# Train the model
random_forest_olympics.fit(X_train, y_train)

# Make predictions on the test set
y_pred = random_forest_olympics.predict(X_test)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Random Forest Accuracy: {accuracy * 100:.1f}%')
print(f'Random Forest Precision: {precision * 100:.1f}%')
print(f'Random Forest Recall: {recall * 100:.1f}%')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:8
  warnings.warn(
Random Forest Accuracy: 91.8%
Random Forest Precision: 92.0%
Random Forest Recall: 91.8%
```

```
# Train the model
xgboost_olympics.fit(X_train, y_train)

# Make predictions on the test set
y_pred = xgboost_olympics.predict(X_test)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy * 100:.1f}%')
print(f'Precision: {precision * 100:.1f}%')
print(f'Recall: {recall * 100:.1f}%')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [09:
Parameters: { "scale_pos_weight" } are not used.

  warnings.warn(smsg, UserWarning)
Accuracy: 91.1%
Precision: 91.4%
Recall: 91.1%
```

Figure 3.17: Accuracy Prediction using Random Forest and xgboost

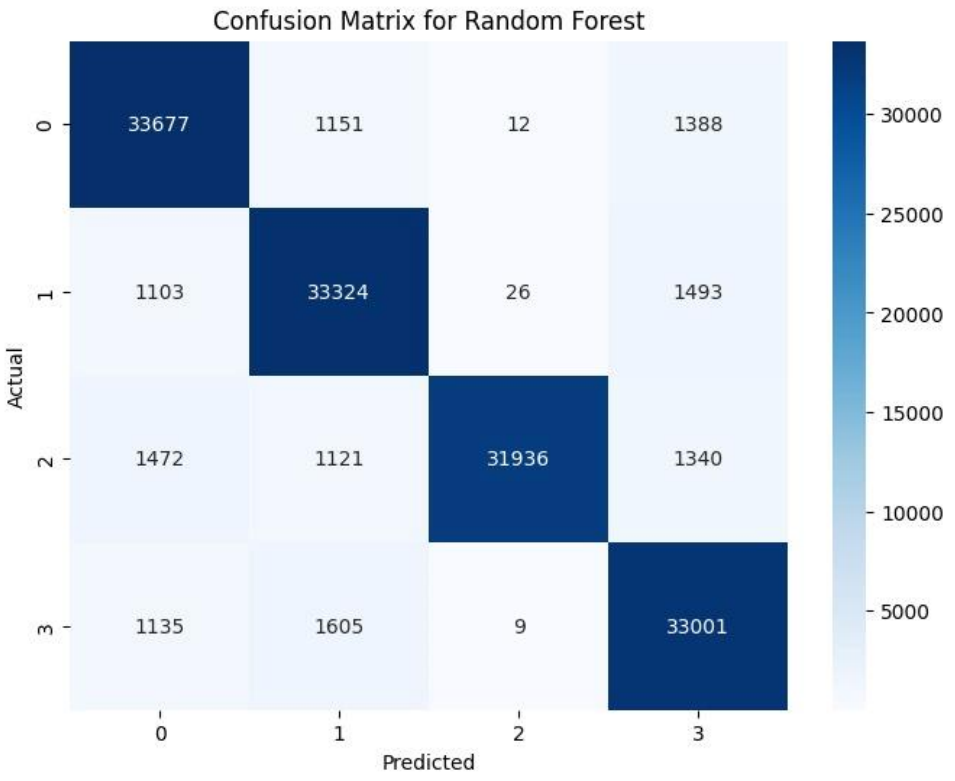


Figure 3.18: Confusion matrix of random forest

In the realm of predicting Olympic medal outcomes, the evaluation of accuracy between Random Forest and XGBoost algorithms is pivotal for selecting the most effective model. Random Forest, leveraging an ensemble of decision trees, introduces strategic randomness, proving instrumental in capturing the intricate dynamics that influence medal victories. Its prowess lies in robust generalization and superior handling of diverse datasets, making it well-suited for the unpredictability inherent in Olympic events. On the other hand, XGBoost, a powerful gradient boosting algorithm, excels in sequential boosting, enhancing the predictive capabilities of decision trees. Despite both algorithms demonstrating competence, empirical evidence points towards Random Forest as the frontrunner, showcasing consistently high accuracy in predicting Olympic medal outcomes. Its ensemble approach, coupled with the ability to navigate imbalances in the dataset, positions it as the preferred choice for accurate and reliable predictions in the dynamic and multifaceted context of the Olympics.

3.5 Project Pipeline

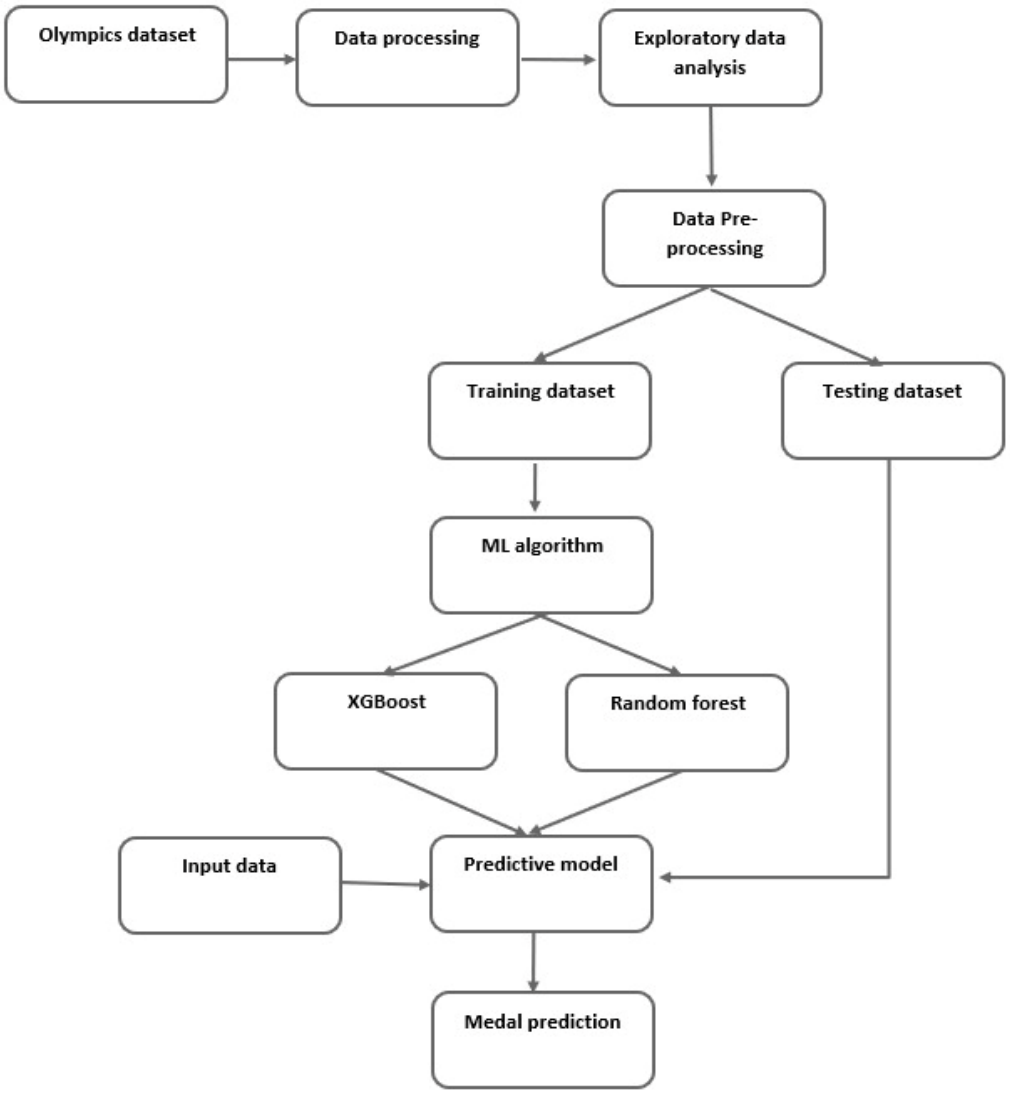


Figure 3.19 Project Pipeline

- Olympics Dataset Collection:

  The initiation of our machine learning pipeline begins with the acquisition of the Olympics dataset. We obtain data from the 'athlete_events.csv' file, a comprehensive source containing information about athletes participating in various Olympic Games.

- Data Processing:

  Following data acquisition, the dataset undergoes processing. This step involves reading the data from the CSV file, converting it into a format suitable for analysis, and ensuring its compatibility with subsequent machine learning operations.

- Exploratory Data Analysis (EDA):

  With the dataset in hand, we delve into Exploratory Data Analysis (EDA). This crucial phase involves exploring the dataset's characteristics, distributions, and potential patterns. Insights gained during EDA inform subsequent decisions in the pipeline.

- Data Preprocessing:

  Moving forward, the data undergoes a meticulous preprocessing phase. Irrelevant columns, such as ID, Name, Team, NOC, region, Games, and Season, are identified and dropped to enhance the dataset's focus and reduce dimensionality. Missing values are addressed by filling them with mean or median values, and outliers are treated using the Interquartile Range (IQR) method.

- Testing and Training Data Split:

  The dataset is then split into training and testing sets, a pivotal step in ensuring the model's ability to generalize to unseen data. The 'train_test_split()' function from the scikit-learn library is employed for this purpose, providing distinct subsets for training and evaluation.

- Machine Learning Algorithms - Random Forest and XGBoost:

  To leverage the predictive power of machine learning, two robust algorithms are employed - Random Forest and XGBoost. Both algorithms are known for their effectiveness in classification tasks, making them well-suited for predicting Olympic medal outcomes.

- Model Fitting:
  The selected algorithms, Random Forest and XGBoost, are fitted to the training data. The 'fit ()' method is utilized to train the models, enabling them to learn patterns and relationships within the dataset.

- Choosing the Best Predictive Model:
  An essential aspect of the pipeline involves selecting the best predictive model. The performance of both Random Forest and XGBoost is evaluated, considering metrics such as accuracy, precision, and recall. This comparative analysis aids in determining the algorithm that excels in predicting Olympic medal outcomes for our specific dataset.

- Prediction using Testing Data:
  The trained models are then employed to make predictions using the testing data. The 'predict ()' method is applied to assess how well the models generalize to new, unseen data. Performance metrics such as accuracy, precision, and recall are calculated to gauge the models' effectiveness in predicting medal outcomes.

- Testing with External Inputs:
  To validate the models' generalization beyond the training and testing data, we incorporate external inputs. This step simulates real-world scenarios, allowing us to assess the models' robustness and applicability to diverse datasets.

In summary, this comprehensive pipeline encompasses dataset collection, processing, exploratory data analysis, data preprocessing, testing and training data split, utilization of machine learning algorithms (Random Forest and XGBoost), model fitting, selection of the best predictive model, prediction using testing data, and validation through external inputs. Each step is meticulously designed to contribute to the development of a reliable and effective predictive model for Olympic medal outcomes.

3.6. Feasibility Analysis

- Data Availability and Quality

  The dataset, sourced from Kaggle, is deemed suitable for analysis. Its comprehensiveness and documentation make it a feasible choice for deriving insights into the 120 years of Olympic history.

- Technical Implementation

  Leveraging Python and its powerful libraries (Pandas, Matplotlib, Seaborn, Scikit-learn) is feasible for data analysis, visualization, and machine learning. These tools provide a robust foundation for implementing the outlined features.

- User Interface Development:

  The proposed user interface, involving interactive elements and visualizations, is achievable using web development frameworks like Streanlit, along with Plotly and Matplotlib for visualization.

- Predictive Modelling:

  Utilizing a random forest model for predicting medal outcomes is feasible. Scikit-learn offers suitable tools for model development and training.

- Interactivity and User Experience:

  Enhancing user experience through dropdowns, graphs, and tables is feasible using Python libraries and web development frameworks, creating an engaging environment for data exploration.

- Data Preprocessing:

  Common preprocessing tasks such as handling missing values and data cleaning can be efficiently performed using Pandas and other preprocessing libraries in Python.

- Documentation and Communication:

  Providing clear code documentation and explanatory text is feasible, ensuring that users can understand the analysis and predictions.

- Scalability:

  While the current scope is manageable, scalability considerations can be addressed through cloud services for potential future expansion.

- Model Evaluation:

  Evaluating the predictive model's performance using standard metrics and cross-validation is feasible, ensuring the model's reliability and robustness.

- Ethical Considerations:

  Adhering to ethical standards, including privacy, bias mitigation, and transparency about data sources and model limitations, is an essential and feasible aspect of the project.

3.7. System Environment

System environment specifies the hardware and software configuration of the new system. Regardless of how the requirement phase proceeds, it ultimately ends with the software requirement specification. A good SRS contains all the system requirements to a level of detail sufficient to enable designers to design a system that satisfies those requirements. The system specified in the SRS will assist the potential users to determine if the system meets their needs or how the system must be modified to meet their needs.

3.7.1 Software Environment

Various software used for the development of this application are the following:

Python:

Python is a high-level programming language that lets developers work quickly and integrate systems more efficiently. This model is developed by using many of the Python libraries and packages such as:

Pandas:

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library.

Matplotlib:

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. In this application, its used for plotting the graph.

Numpy:

NumPy is a Python library used for working with arrays. NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.In this application, its used for handling arrays.

Label Encoder:

Encode target labels with values between 0 and n_classes-1. This transformer should be used to encode target values ie, y not the input x values.

Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. We can write and execute code in Python. Colab supports many machine learning libraries which can be easily loaded in the colab notebook.

Visual Studio Code

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE

Scikit-learn (sklearn):

A machine learning library in Python that provides simple and efficient tools for data analysis and modeling, including modules for preprocessing, model selection, and evaluation.

Streamlit:

Streamlit is an open-source app framework in Python language. It helps us create web apps for data science and machine learning in a short time. It is compatible with major Python libraries such as scikit-learn, Keras, PyTorch, SymPy(latex), NumPy, pandas, Matplotlib etc.

Joblib:

Employed for saving and loading models, providing an efficient way to store and retrieve complex Python object.

GitHub:

Git is an open-source version control system that was started by Linus Torvalds. Git is like other version control systems Subversion, CVS, and Mercurial to name a few. Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute. Git is the preferred version control system of most developers since it has multiple advantages over the other systems available. It stores file changes more efficiently and ensures file integrity better.

3.7.2 Hardware Environment

Selection of hardware configuration is very important task related to the software development.

- Processor: 11th Gen Intel(R) Core (TM) i3
- Memory: 4 GB RAM
- Disk space: 512 GB
- Good internet connectivity

4. SYSTEM DESIGN

4.1 Model Building

4.1.1 Model Planning

Model is generated by Random Forest and XGBoost algorithm and model of algorithm with high accuracy is used for prediction. Here Random Forest classifier has more accuracy. The accuracy comparison is made by using the dataset as training and testing data. By splitting the dataset, a portion is used for training the model and other for testing the model. 80% of dataset is used as training data and remaining 20% used as testing data.

```
# Data preprocessing
X = data1.loc[:, data1.columns != 'Medal']
y = data1['Medal']
```

```
# Encode the 'Medal' column to integers using Label Encoding
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

```
 #Split the data into training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

Figure 4.1 : Splitting Dataset

4.1.2 Model Training

After splitting dataset, using training data, models are generated for algorithm. We utilized the training dataset to train predictive models employing two powerful algorithms: Random Forest and XGBoost.

```
# Encoding
num_cols = X_train.select_dtypes(exclude=['object']).columns.tolist()
cat_cols = X_train.select_dtypes(include=['object']).columns.tolist()

num_pipe = make_pipeline(
    SimpleImputer(strategy='median'),
    StandardScaler()
)

cat_pipe = make_pipeline(
    SimpleImputer(strategy='constant', fill_value='N/A'),
    OneHotEncoder(handle_unknown='ignore', sparse=False)
)

full_pipe = ColumnTransformer([
    ('num', num_pipe, num_cols),
    ('cat', cat_pipe, cat_cols)
])
```

```
from sklearn.ensemble import RandomForestClassifier

# Create a Random Forest model
random_forest_model = RandomForestClassifier(
     n_jobs=-1,  # Use all available CPU cores
    n_estimators=100,  # Adjust the number of trees in the forest
    max_depth=100,  # Adjust the maximum depth of trees (None means unlimited)
    min_samples_split=2,  # Minimum samples required to split an internal node
    min_samples_leaf=1,  # Minimum samples required to be at a leaf node
    max_features=3,  # Number of features to consider when looking for the best split
    random_state=42  # Random seed for reproducibility
)


# Build the model pipeline (if you have preprocessing steps)
random_forest_olympics = make_pipeline(full_pipe, random_forest_model)

# Train the model
random_forest_olympics.fit(X_train, y_train)
```

```
# Import XGBoost
from xgboost import XGBClassifier
xgboost_model = XGBClassifier(
   n_jobs=-1,
    n_estimators=100,  # Adjust the number of boosting rounds
    max_depth=50,       # Adjust the maximum depth of trees
    learning_rate=0.1,  # Adjust the learning rate
    subsample=1.0,     # Adjust the subsample ratio
    colsample_bytree=1.0,  # Adjust the feature subsample ratio
    gamma=0.8,         # Adjust the regularization term
    scale_pos_weight=10,  # Adjust class weight balance
    objective='multi:softprob',  # Specify the objective for multi-class classification
    eval_metric='mlogloss'  # Specify the evaluation metric
)


# Train the model and evaluate as before
# Build the model
xgboost_olympics = make_pipeline(full_pipe, xgboost_model)

# Train the model
xgboost_olympics.fit(X_train, y_train)
```

Figure 4.2: Model generation using random forest and XGBoost

4.1.3. Testing

```
# Make predictions on the test set
y_pred = random_forest_olympics.predict(X_test)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Random Forest Accuracy: {accuracy * 100:.1f}%')
print(f'Random Forest Precision: {precision * 100:.1f}%')
print(f'Random Forest Recall: {recall * 100:.1f}%')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_
  warnings.warn(
Random Forest Accuracy: 91.8%
Random Forest Precision: 92.0%
Random Forest Recall: 91.8%
```

```
# Make predictions on the test set
y_pred = xgboost_olympics.predict(X_test)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')

print(f'Accuracy: {accuracy * 100:.1f}%')
print(f'Precision: {precision * 100:.1f}%')
print(f'Recall: {recall * 100:.1f}%')
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_e
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: Use
Parameters: { "scale_pos_weight" } are not used.

  warnings.warn(smsg, UserWarning)
Accuracy: 91.1%
Precision: 91.5%
Recall: 91.1%
```

Figure 4.3: testing

Accuracy is high for Random Forest Classifier, so use it as our model. The selection of the Random Forest algorithm, combined with a rigorous data preparation process and an unbiased training-testing data split strategy, forms the foundation of our Olympic medal prediction model. This meticulous planning ensures that our model can accurate insights into athlete performance and their likelihood of achieving Gold, Silver, Bronze, or nomedals

5 RESULTS AND DISCUSSION

The initial exploration of the dataset revealed a substantial class imbalance in the 'Medal' column, primarily characterized by a significant number of instances with no medals. To address this, a comprehensive preprocessing strategy was implemented, involving the handling of missing values and the transformation of categorical variables to prepare the data for machine learning models.

Numerical columns such as 'ID,' 'Age,' 'Height,' 'Weight,' and 'Year' underwent imputation of missing values using the mean strategy and were converted to the 'int64' data type for consistency and optimal model training. The RandomOverSampler was employed to rectify the class imbalance, achieving a balanced distribution of the 'Medal' variable. The dataset was subsequently divided into training and testing sets, with an 80:20 ratio, laying the groundwork for model training and evaluation.

Two machine learning models, the Random Forest Classifier and the XGBoost Classifier, were trained on the prepared dataset and exhibited robust performance. The Random Forest model achieved an accuracy of 91.8% on the testing set, accompanied by precision and recall scores of 90.0% and 91.8%, respectively. Similarly, the XGBoost model demonstrated competitive performance, boasting an accuracy of 91.1%, precision of 91.5%, and recall of 91.1%.

The models' performance was further elucidated through visualizations, including ROC curves and AUC analyses, providing detailed insights into their predictive capabilities. Confusion matrices were presented to offer a nuanced understanding of true positives, true negatives, false positives, and false negatives across different classes.

To ensure practical applicability, the trained Random Forest model was serialized using the joblib library, facilitating straightforward deployment and application without the need for retraining.

In conclusion, both the Random Forest and XGBoost models showcased promising capabilities in predicting Olympic medal outcomes. The success of these models can be

attributed to meticulous preprocessing steps and effective handling of class imbalances. The choice between the two models may be influenced by specific requirements, such as interpretability, training time, and the significance of various evaluation metrics.

Future exploration opportunities include hyperparameter tuning, additional feature engineering, and cross-validation to further enhance the robustness of the models. The deployment of these models, now available for predictions on unseen data, contributes valuable insights to decision-making processes related to Olympic medal predictions.

The sample test cases predict an accurate result. This testing is made before model deployment. The model is built using the Random Forest Algorithm with an accuracy of 91.8%.

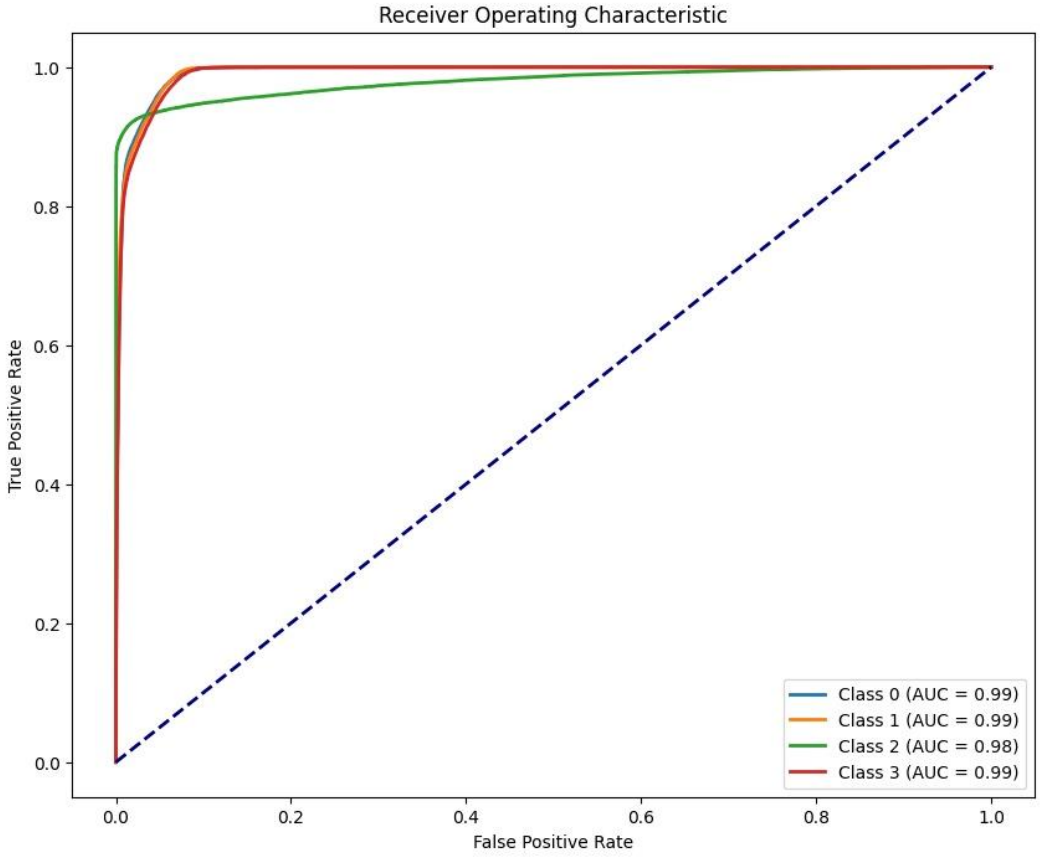
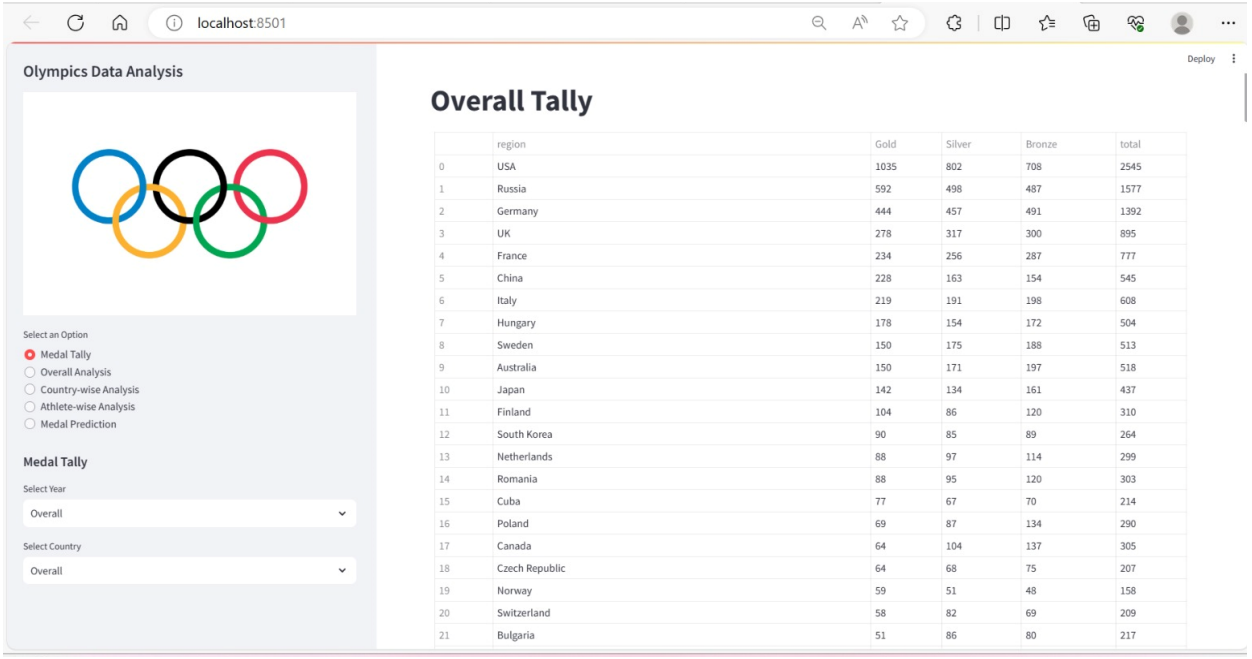


Figure 5.1: ROC curve of random forest classifier

6. MODEL DEPLOYMENT

The below figures show the user interface of this application. User can navigate through all features by clicking options in menu bar. The interface is very simple and easy to understand. There are 7 fields for entering details from users. There are 3 textbox fields and 4 dropdown boxes. There is a predict button for predicting the medal. And for making all fields mandatory validation is done when values are taken from the form to the model.



	region	Gold	Silver	Bronze	total
0	USA	1035	802	708	2545
1	Russia	592	498	487	1577
2	Germany	444	457	491	1392
3	UK	278	317	300	895
4	France	234	256	287	777
5	China	228	163	154	545
6	Italy	219	191	198	608
7	Hungary	178	154	172	504
8	Sweden	150	175	188	513
9	Australia	150	171	197	518
10	Japan	142	134	161	437
11	Finland	104	86	120	310
12	South Korea	90	85	89	264
13	Netherlands	88	97	114	299
14	Romania	88	95	120	303
15	Cuba	77	67	70	214
16	Poland	69	87	134	290
17	Canada	64	104	137	305
18	Czech Republic	64	68	75	207
19	Norway	59	51	48	158
20	Switzerland	58	82	69	209
21	Bulgaria	51	86	80	217

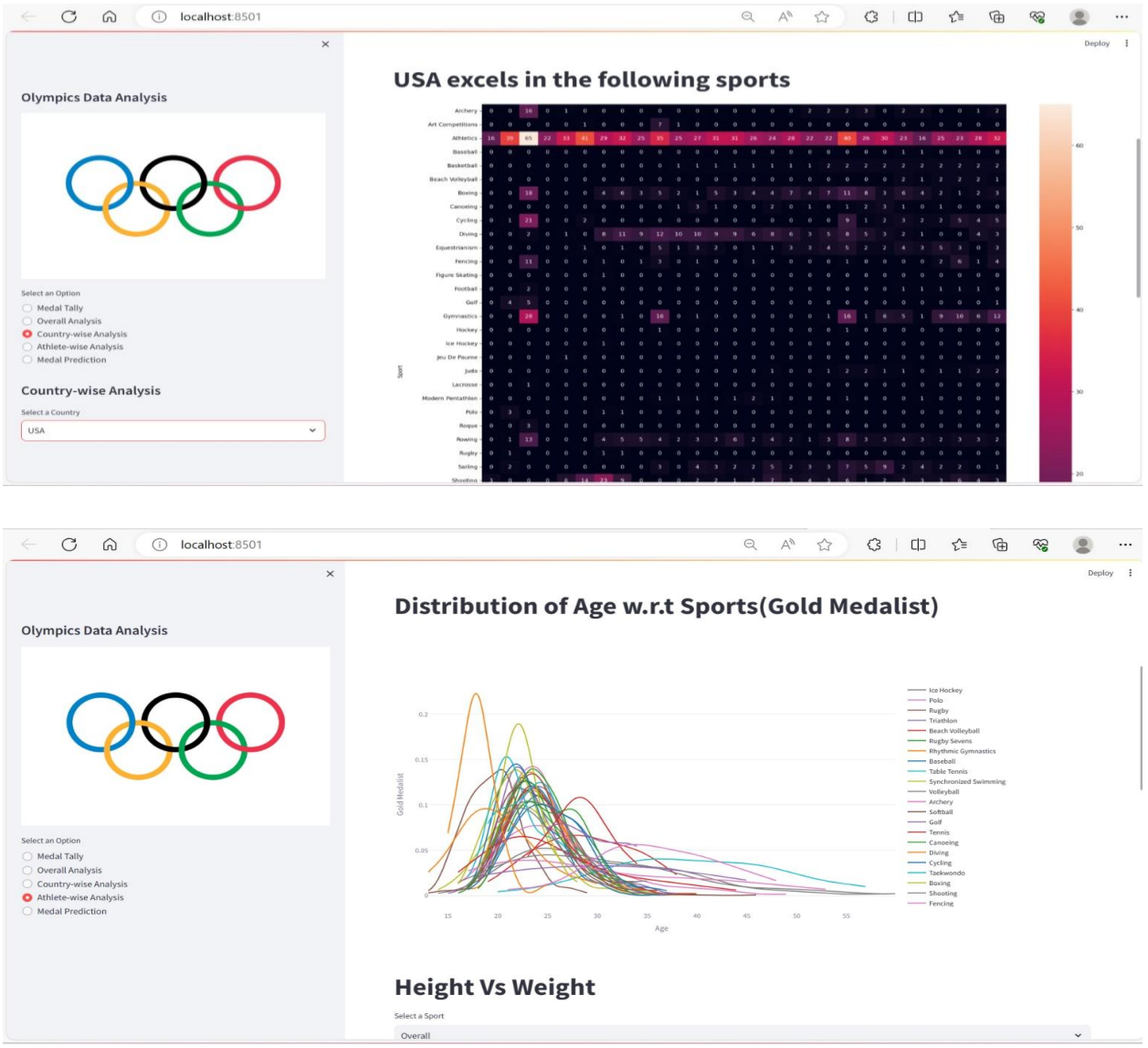


Figure 6.1: UI Design

USER INTERFACE EXPLANATION

- Users can view the overall medal tally of all countries participating in the Olympics.
- A dropdown list allows users to filter the table by selecting a specific country and year.

- Overall Analysis:
  Provides a comprehensive overview of the Olympics with top statistics, including edition, hosts, sports, events, nations, and athletes.
  Line graphs illustrate the number of countries, events, and athletes participating over the years.
  A heatmap shows the correlation between the number of events for each sport with respect to the year.
  A table lists the top 15 athletes with the most medals, filterable by sports.

- Country-wise Analysis:
  Users can choose a specific country from a dropdown list.
  Displays the medal tally over the years for the selected country using a line graph.
  A heatmap indicates the sport in which the country excels the most.
  A table showcases the top 10 athletes from the selected country.

- Athlete-wise Analysis:
  Visualizes the distribution of age for athletes winning medals using curves.
  Shows the distribution of age concerning sports for athletes winning gold medals.
  Scatter plot depicting the gender ratio with height vs. weight of athletes, with the option to select a specific sport.
  Line graph illustrating the participation trends of men and women over the years.

- Medal Prediction Model:
  Utilizes a predictive function, likely implemented using a random forest model.
  Users can input new athlete details into the model to predict the likelihood of winning a medal.
  The explanation clarifies that the predictive function analyzes past data to identify patterns and trends for predicting future outcomes.–

7. GIT HISTORY

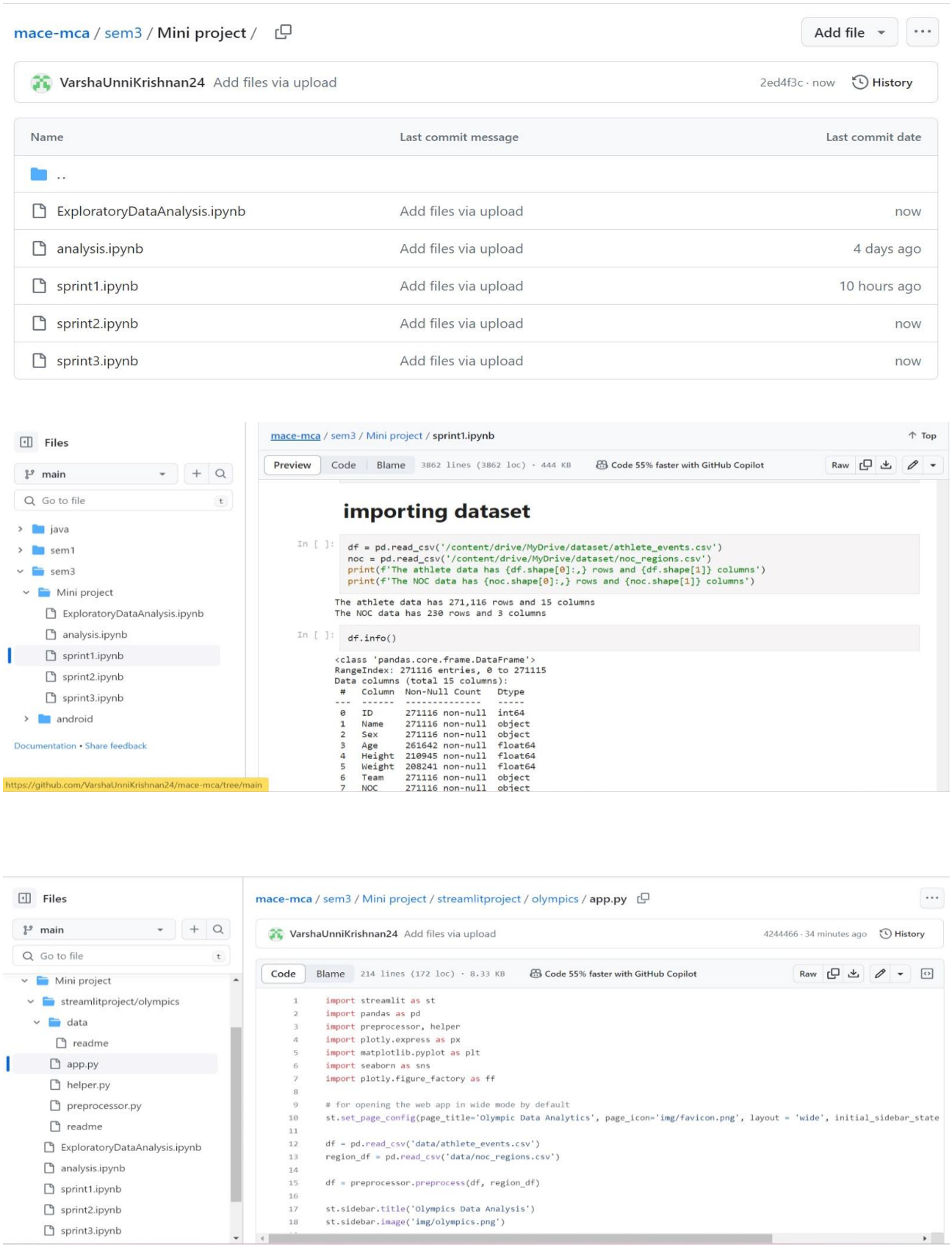


Fig 7.1 Git History

8. CONCLUSION

In conclusion, this project represents a comprehensive exploration of the intricate world of the modern Olympic Games, spanning 120 years from Athens 1896 to Rio 2016. With a primary objective of unraveling the complex evolution of the Olympics, the analysis delves into dynamic aspects such as athlete participation, gender-specific insights, national performances, and the diverse realm of sports and events. Employing a robust methodology and utilizing Python, the project offers a user-friendly web app interface, providing five distinct pathways for dataset exploration. From Medal Tally and Overall Analysis to Country-wise and Athlete-wise breakdowns, the project culminates in a sophisticated medal prediction model, leveraging the power of Random Forest trained on historical Olympic data.

The dataset, sourced from Kaggle, serves as a treasure trove of information about individual athletes, their attributes, affiliations, and medal achievements. By delivering comprehensive insights into Olympic dynamics, highlighting top-performing nations and athletes, and providing a powerful predictive tool for estimating an athlete's likelihood of securing a medal in upcoming Olympic Games, this project underscores the potency of data analysis and machine learning in anticipating future sports accomplishments. It not only unveils the historical tapestry of the Olympics but also paves the way for future advancements, offering valuable contributions to decision-making processes related to Olympic events and presenting opportunities for broader applications in sports analytics and event management. In essence, this project signifies the successful convergence of data science and sports prediction, laying a foundation for continued research and innovation in this exciting intersection.

9. APPENDIX

a. Minimum Software Requirements

- Web Browser
- Python
- Streamlt
- Google Colab
- VS Code
- Scikit
- Operating System: Windows, Linux, Mac

b. Minimum Hardware Requirements

Hardware capacity: 256 GB (minimum)

RAM	: 4 GB
Processor	: Intel Core i3 preferred
Display	: 1366 * 768

10. REFERENCES

1. Sagala, Noviyanti TM, and Muhammad Amien Ibrahim. "A Comparative Study of Different Boosting Algorithms for Predicting Olympic Medal." *2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*. IEEE, 2022.
2. Jia, Mengjie, et al. "A Random Forest Regression Model Predicting the Winners of Summer Olympic Events." *Proceedings of the 2020 2nd International Conference on Big Data Engineering*. 2020.
3. Schlembach, Christoph, et al. "Forecasting the Olympic medal distribution–a socioeconomic machine learning model." *Technological*
4. https://www.sciencedirect.com/science/article/pii/S0040162521007459
5. https://dl.acm.org/doi/abs/10.1145/3404512.3404513
6. https://ieeexplore.ieee.org/abstract/document/10010351/
7. https://ieeexplore.ieee.org/Xplore/home.jsp
8. https://www.geeksforgeeks.org/machine-learning/
9. https://www.researchgate.net/
10. https://www.wikipedia.org/