

Data Mining & Visualization——The Olympic Dataset

Xiyu Han 2300244

Contents

1 Introduction of Olympic Data	2
1.1 Background	2
1.2 Dataset characteristics	3
2 Related work	4
2.1 WEKA	4
2.2 Data mining	5
3 Initial Data Pre-Processing and Visualization	5
3.1 Data description and visualization	5
3.2 Data Pre-Processing	11
4 Data Analysis with Machine Learning	12
Question1——Can we predict whether an athlete will win a medal based on features like age, height, weight, team, and sport?	13
4.1 Random Forest Algorithm (Python)	13
4.2 XGBoost (Python)	16
4.3 RandomTree (WEKA)	19
Question2——How has the number of participating athletes from each country changed over approximately 100 years, from the 1910s to the 2010s? (Tableau)	21
Question3——How has the number of events in the Winter and Summer Olympics changed over the years? (Python)	23
Question4——How does the proportion of medal colors vary among the top 30 countries with the most medals? (Python)	24
5 Challenge and Reflection	25
5.1 Challenge	25
5.2 Learned	25
5.3 Reflection and Future	26
6 Conclusion	26

Data Mining & Visualization——The Olympic Dataset

Xiyu Han 2300244

Abstract

This report aims to analyze a Olympic dataset to uncover significant patterns and trends. Prior to analysis, the dataset will undergo a series of preprocessing steps to ensure compatibility with leading data mining and visualization tools such as Weka and Tableau. Once imported, a variety of analytical methods will be employed to develop models and visualizations, thereby facilitating the identification of key patterns. The analysis will be guided by predefined questions, with the goal of using the models to provide answers.

1 Introduction of Olympic Data

1.1 Background

The Olympic Games are an international multi-sport event held every four years in which thousands of athletes from around the world participate in various sports competitions. The Olympics are one of the most significant and prestigious sporting events globally, promoting unity, friendship, and fair play among nations. This dataset, provided by BHANUPRATAP BISWAS and sourced from Kaggle (<https://www.kaggle.com/datasets/bhanupratapbiswas/olympic-data/data>), provides a comprehensive historical record of this major international multi-sport event, held every four years. It encompasses detailed information on the events, athletes, and medal tallies, offering valuable insights into the performances and trends over the years. This dataset serves as a rich resource for analysis and research, reflecting the extensive participation of thousands of athletes from around the world in a wide range of sports competitions.

Analyzing historical Olympic Games data holds significant value for multiple reasons. It allows for the identification of performance trends among athletes and countries, aiding in strategic planning and decision-making for training programs and

resource allocation to enhance competitiveness in future events. Moreover, understanding how technological advancements and rule changes affect competition outcomes can guide appropriate modifications to sporting regulations. Thus, this data not only deepens our understanding of Olympic history and development but also supports a wide range of practical applications from policy formulation to training methodologies.

1.2 Dataset characteristics

This dataset includes 70,000 data and 15 attributes, and it is structured in a user-friendly CSV format, making it accessible for various types of analysis. It includes key attributes such as athlete names, nationalities, sports, events, and results, along with more detailed information like age, height, weight, and the medals won. This allows us to dive deep into the specifics of Olympic performance and trends.

Table 1-1. Olympic Dataset

Key	Value
ID	Unique number for each athlete
Name	Athlete’s name
Sex	M or F
Age	Integer
Height	In centimeters
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

2 Related work

2.1 WEKA

WEKA is a popular suite of machine learning software written in Java, developed by the University of Waikato in New Zealand. It provides a comprehensive collection of tools for data preprocessing, classification, regression, clustering, association rules, and visualization. Weka is designed to be accessible to those new to data mining but also powerful enough for more experienced practitioners. It is widely used in academia and industry for teaching, research, and practical applications due to its user-friendly interface and its extensive variety of algorithms that can be applied directly to a dataset or called from custom Java code. Weka also supports several standard data mining tasks, more specifically, data processing and modeling through its software environment.

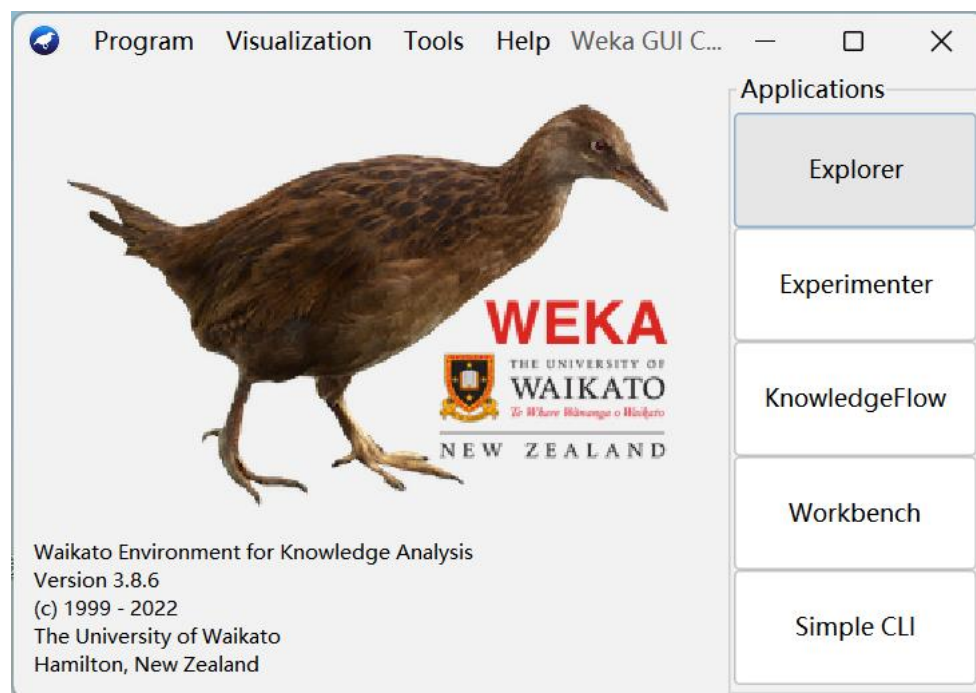


Figure 2-1. WEKA

2.2 Data mining

Data mining is the process of discovering patterns, correlations, and insights from large sets of data using computational techniques. It involves methods at the intersection of machine learning, statistics, and database systems. Tools like Weka, Python, and Tableau are integral to facilitating these tasks across different stages of data analysis. Weka, with its robust suite of machine learning algorithms, is excellent for experimenting with data and performing various data mining tasks directly through its GUI or via Java code. Python, with libraries such as pandas, scikit-learn, and TensorFlow, offers a versatile scripting environment for more customized data mining, machine learning, and deep learning applications. Meanwhile, Tableau excels in the visualization of data, enabling users to create intuitive and interactive dashboards that reveal trends and patterns essential for making informed decisions. Collectively, these tools provide a powerful arsenal for data miners, blending sophisticated data analysis capabilities with user-friendly interfaces to explore and interpret vast datasets effectively.

3 Initial Data Pre-Processing and Visualization

3.1 Data description and visualization

(1) Import the dataset and view the data:

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	35656	Stuart Fitzsimmons	M	19.0	NaN	NaN	Great Britain	GBR	1976 Winter	1976	Winter	Innsbruck	Alpine Skiing	Alpine Skiing Men's Giant Slalom	NaN
69996	35656	Stuart Fitzsimmons	M	19.0	NaN	NaN	Great Britain	GBR	1976 Winter	1976	Winter	Innsbruck	Alpine Skiing	Alpine Skiing Men's Slalom	NaN
69997	35657	David Thomas "Dave" Fitzsimons	M	26.0	170.0	65.0	Australia	AUS	1976 Summer	1976	Summer	Montreal	Athletics	Athletics Men's 10,000 metres	NaN
69998	35657	David Thomas "Dave" Fitzsimons	M	30.0	170.0	65.0	Australia	AUS	1980 Summer	1980	Summer	Moskva	Athletics	Athletics Men's 5,000 metres	NaN
69999	35658	Marek Fiurek	M	23.0	180.0	68.0	Czech Republic	CZE	1998 Winter	1998	Winter	Nagano	Nordic Combined	Nordic Combined Men's Team	NaN

Figure 3-1. View the data

(2) Use the .info() function to obtain the dataset details as follows:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype  
---  --
 0    ID      70000 non-null  int64  
 1   Name     70000 non-null  object  
 2    Sex     70000 non-null  object  
 3    Age     67268 non-null  float64 
 4   Height  53746 non-null  float64 
 5   Weight  52899 non-null  float64 
 6    Team    70000 non-null  object  
 7    NOC     70000 non-null  object  
 8   Games   70000 non-null  object  
 9    Year    70000 non-null  int64  
10   Season  70000 non-null  object  
11   City     70000 non-null  object  
12   Sport    70000 non-null  object  
13   Event    70000 non-null  object  
14   Medal    9690 non-null   object  
dtypes: float64(3), int64(2), object(10)
memory usage: 8.0+ MB
```

Figure 3-2. Data details

At the same time, check for missing value:

```
df.isnull().sum()

ID          0
Name        0
Sex         0
Age        2732
Height     16254
Weight     17101
Team        0
NOC         0
Games       0
Year        0
Season      0
City        0
Sport       0
Event       0
Medal      60310
dtype: int64
```

Figure 3-3. Missing value of dataset

(3) Histograms below show the distributions of age, height, and weight of the athletes in the dataset.

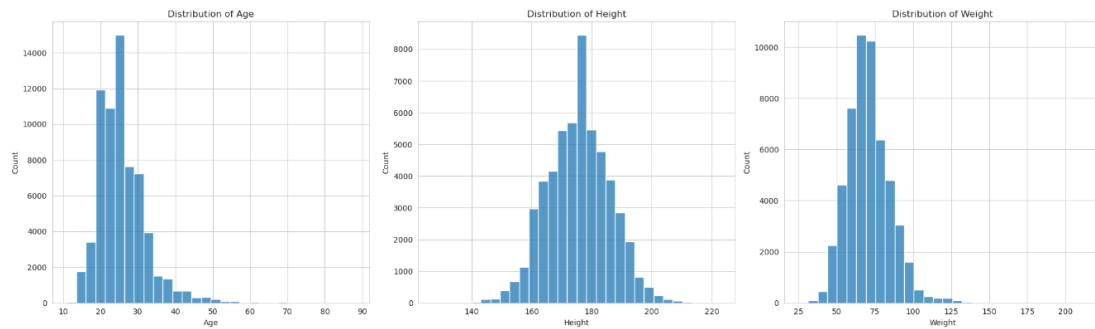


Figure 3-4. Statistical characteristics of age, height, and weight

- ① Age: Most athletes are in their early twenties to late thirties. There are few athletes younger than 20 and older than 40, which is expected given the physical demands of most Olympic sports.
- ② Height: The most common heights for athletes are between 170 cm and 190 cm. There are few athletes shorter than 150 cm or taller than 210 cm.
- ③ Weight: Most athletes weigh between 60 kg and 80 kg. Few athletes weigh less than 40 kg or more than 100 kg.
- (4) Gender Distribution of Athletes: Number of Male is higher than Female.

Rate between Male to Female

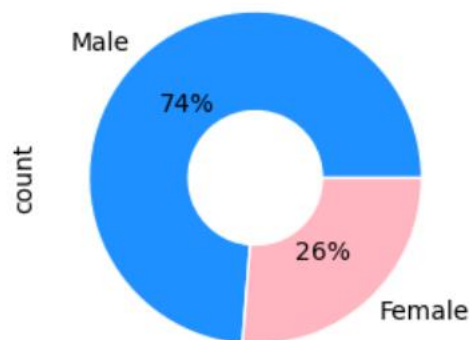


Figure 3-5. Statistical characteristics of gender

(5) Top 10 Teams with the Most Participants:

The United States has the most participants, followed by France, Great Britain, and Italy. This could be due to a variety of factors, including the size of the country, the level of investment in sports, and historical involvement in the Olympics.

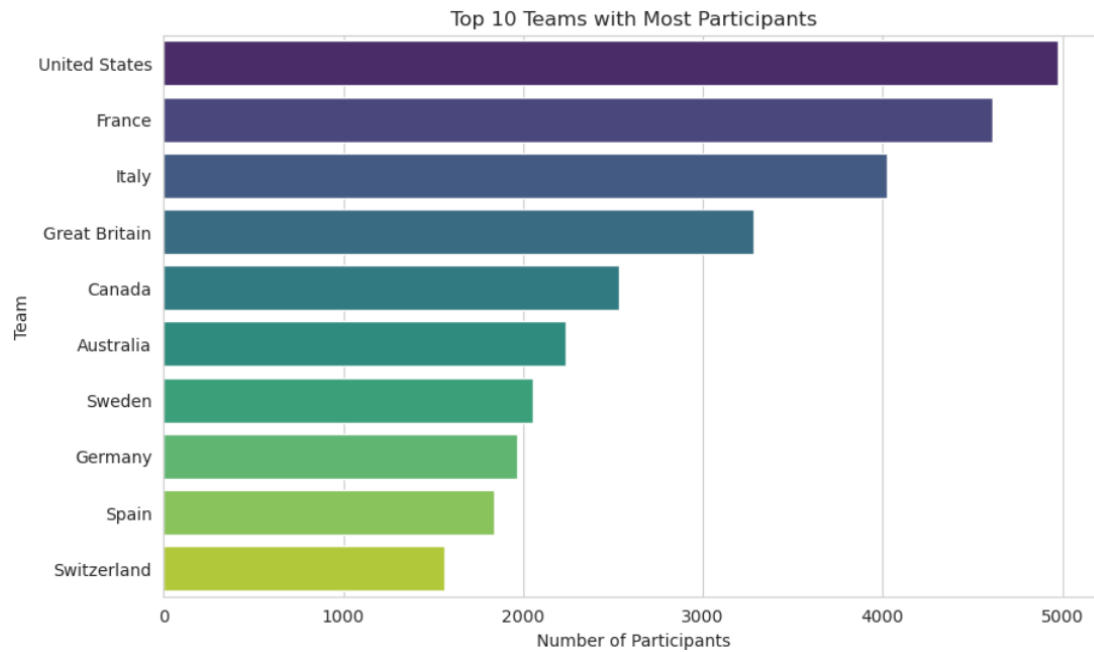


Figure 3-6. Top 10 teams with the most participants

(6) Top 10 Teams with Most Medals:

The bar chart shows the United States significantly leading the Olympic medal count, with over 1400 medals. France and Italy complete the top three.

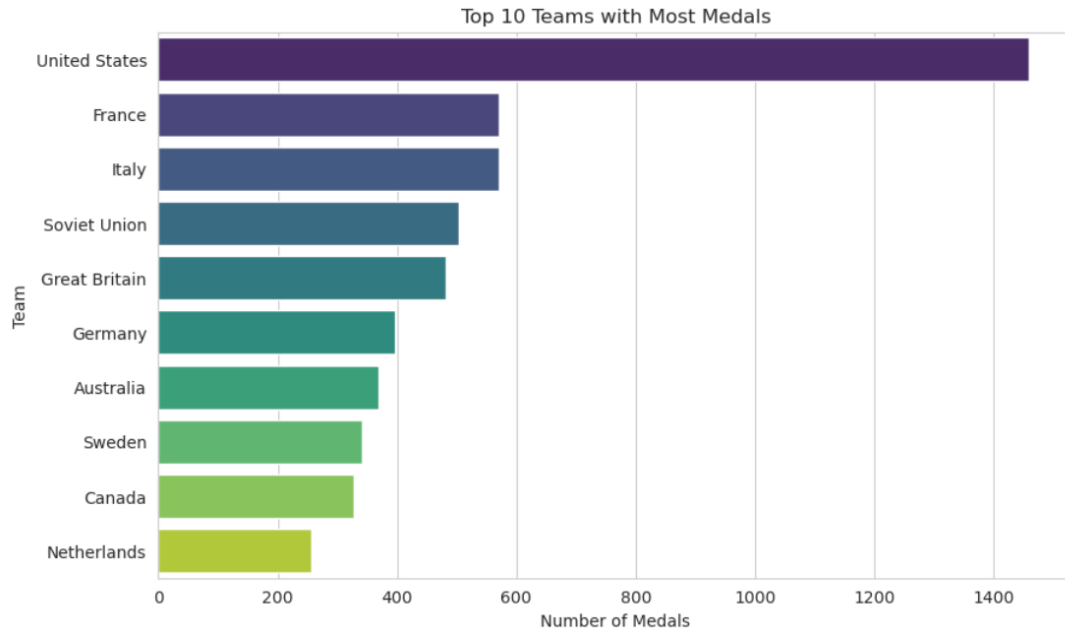


Figure 3-7. Top 10 teams with the most medals

(7) Number of Athletes Per Year:

This line graph illustrates the growth in the number of athletes participating in the Olympics over time, with a general upward trend. Post-1980, there's a visible

oscillation in numbers, reflecting the alternation between Summer and Winter Games.

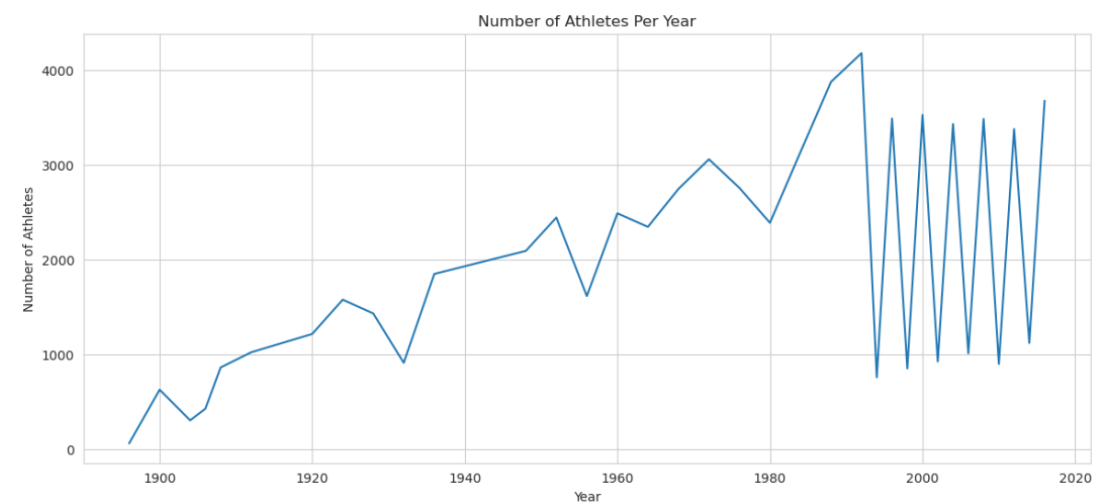


Figure 3-8. Numbers of athletes per year

(8) Top 10 Sports with Most Unique Events:

Athletics tops the chart with the most unique events in Olympic history, followed by sports like shooting and swimming. Cycling and sailing also feature prominently, while gymnastics and rowing have fewer events but still make the top 10 list. This diversity reflects the range of skills and disciplines celebrated in the Olympic tradition.

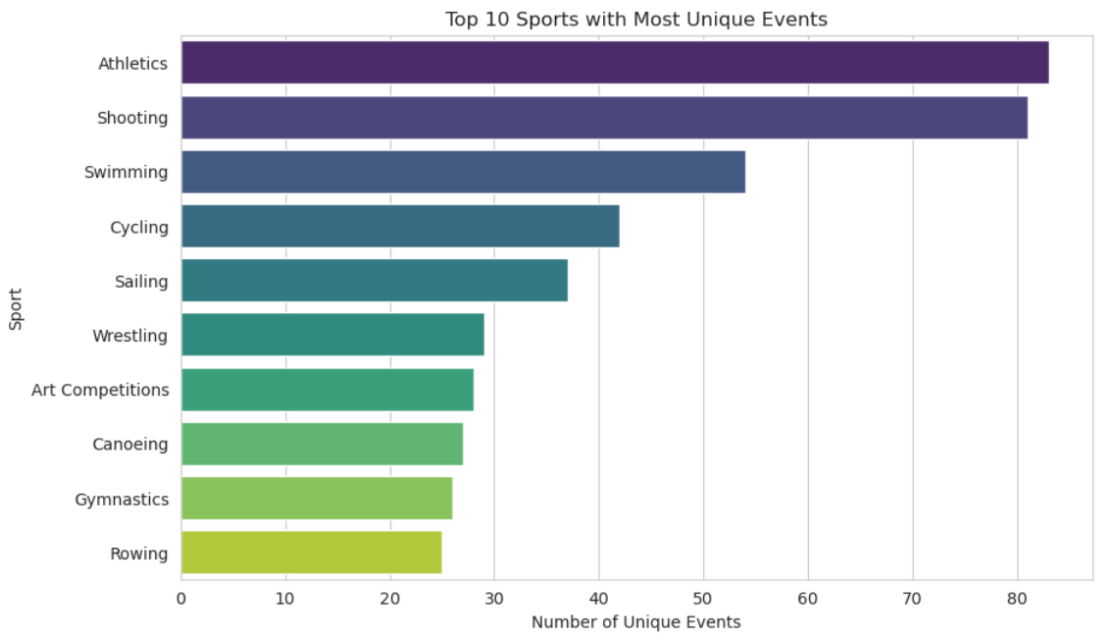


Figure 3-9. Top 10 sports with the most events

(9) Top 10 Sports with Most Athletes:

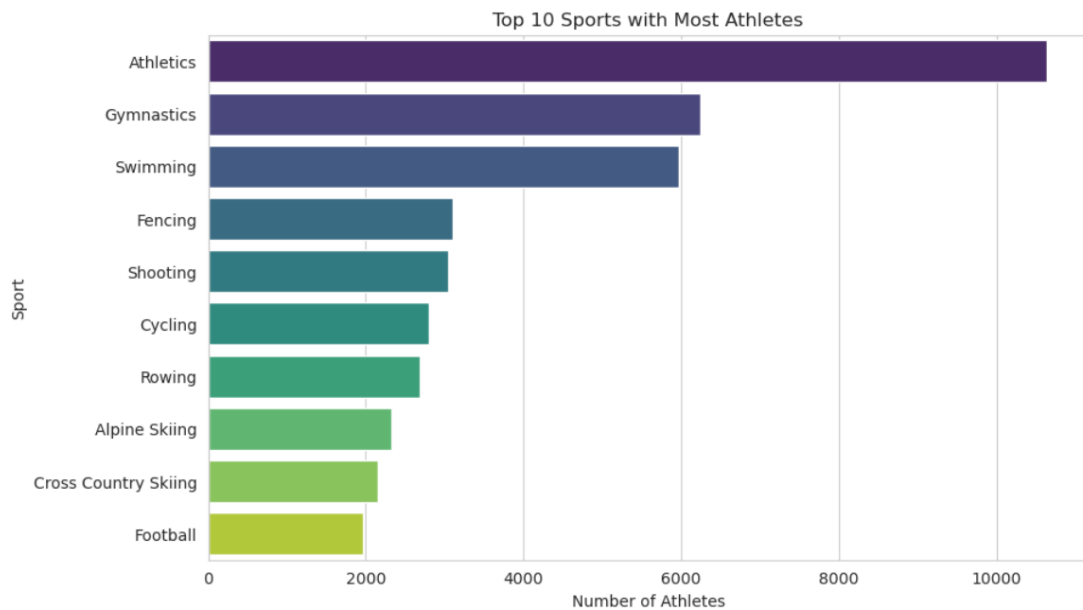


Figure 3-10. Top 10 sports with most athletes

(10) Male and Female Participants in Each Sport

The bar chart displays a significant disparity in the medal distribution by gender, with male athletes (M) participant more than female athletes (F).

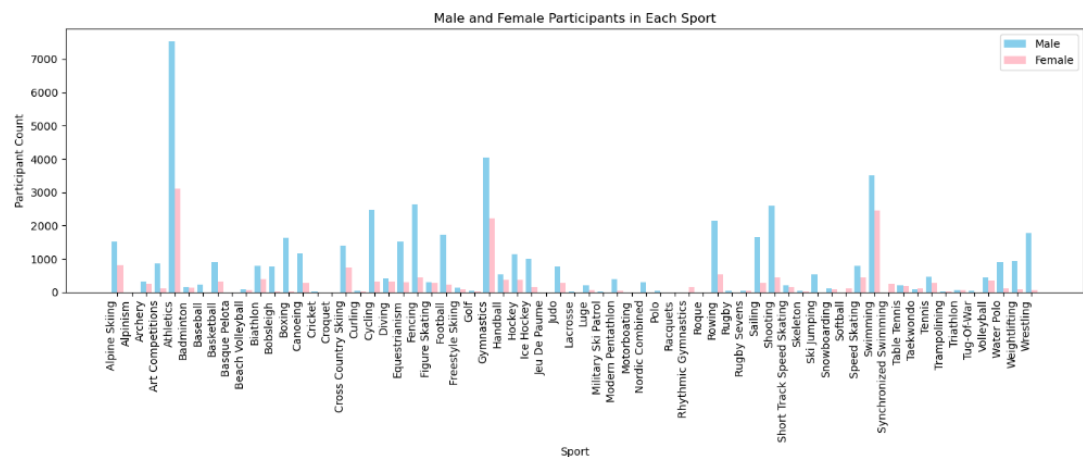


Figure 3-11. Male and female participants in each sport

(11) Relation between the Height and Weight of Athlete with respect to Medals:

The distribution shows a wide range of weights and heights with no clear distinction or clustering by medal type, suggesting that winning different types of medals does not correlate strongly with an athlete's height or weight across various sports.

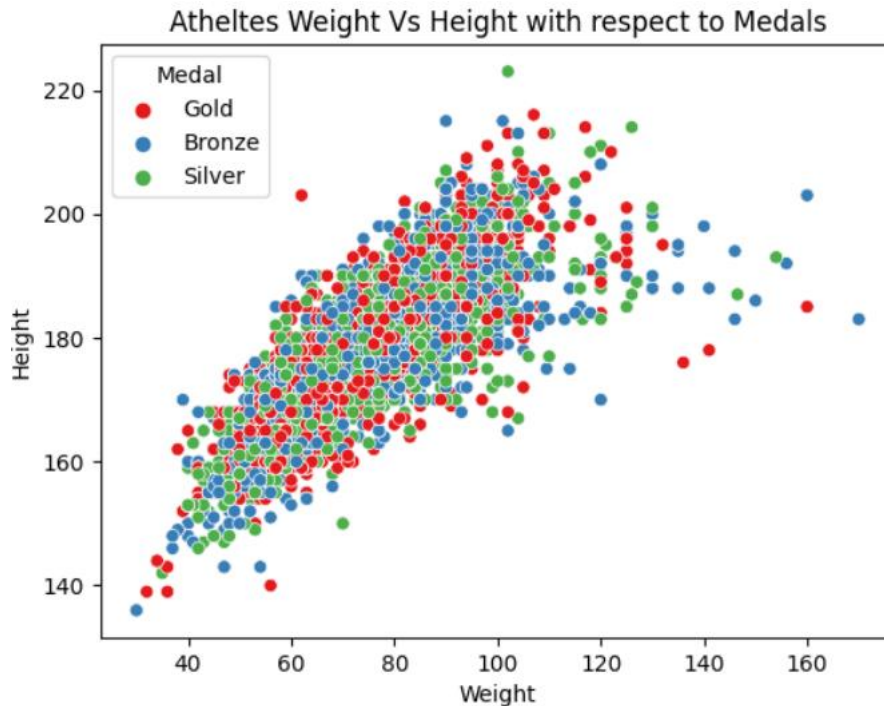


Figure 3-12. Relation between the height and weight of athlete with respect to medals

3.2 Data Pre-Processing

(1) Initially, a copy of the dataset is created to ensure the original data remains unaltered. A new binary column named 'Medal_Won' is then added to indicate whether an athlete has won a medal, transforming any existing medal information into a format suitable for classification algorithms. Subsequently, the 'Medal' column is removed as it becomes redundant after the transformation:

```
# Create a copy of the dataset to avoid modifying the original
data = dataset.copy()

# Create a binary 'Medal_Won' column: 1 if the athlete won a medal, 0 otherwise
data['Medal_Won'] = data['Medal'].notnull().astype(int)

# Drop the 'Medal' column as it's no longer needed
data.drop('Medal', axis=1, inplace=True)
```

Figure 3-13. Process the 'Medal'

(2) Address the missing values in 'Age', 'Height', and 'Weight' by replacing them with the median of their respective columns, which is a common practice to handle gaps in datasets without introducing bias. Categorical features such as 'Sex', 'Team', 'NOC', 'Games', 'Season', 'City', 'Sport', and 'Event' are converted into numerical values using a LabelEncoder, making them interpretable for machine learning models.:

```
# Handle missing values in 'Age', 'Height', and 'Weight' by filling with the median
for column in ['Age', 'Height', 'Weight']:
    data[column].fillna(data[column].median(), inplace=True)

# Convert categorical columns to numeric using LabelEncoder
le = LabelEncoder()
for column in ['Sex', 'Team', 'NOC', 'Games', 'Season', 'City', 'Sport', 'Event']:
    data[column] = le.fit_transform(data[column])
```

Figure 3-14. Fill the miss value

(3) Finally, the dataset is split into features and target labels, and further divided into training and testing sets, providing the necessary structure for supervised learning tasks.:

```
# Split the data into features (X) and target (y)
X = data.drop(['Medal_Won', 'Name'], axis=1)
y = data['Medal_Won']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 3-15. Split the data

4 Data Analysis with Machine Learning

The dataset will be used to investigate and attempt to answer the following four questions:

Question	Technique
1.Can we predict whether an athlete will win a medal based on features like age, height, weight, team, and sport?	Python and WEKA
2.How has the number of participating athletes from each country changed over approximately 100 years, from the 1910s to the 2010s?	Tableau
3.How has the number of events in the Winter and Summer Olympics changed over the years?	Python
4.How does the proportion of medal colors vary among the top 30 countries with the most medals?	Python

Question1——Can we predict whether an athlete will win a medal based on features like age, height, weight, team, and sport?

4.1 Random Forest Algorithm (Python)

The Random Forest algorithm is a form of ensemble learning that aggregates the predictions of multiple decision trees to make more accurate predictions for classification and regression tasks. Here's how it operates: the algorithm constructs each tree using a random subset of training data samples, with the size of this subset being flexible but not exceeding the full training set. For each tree, the algorithm also randomly selects a subset of features, significantly fewer than the total number of available features, to determine the best splits.

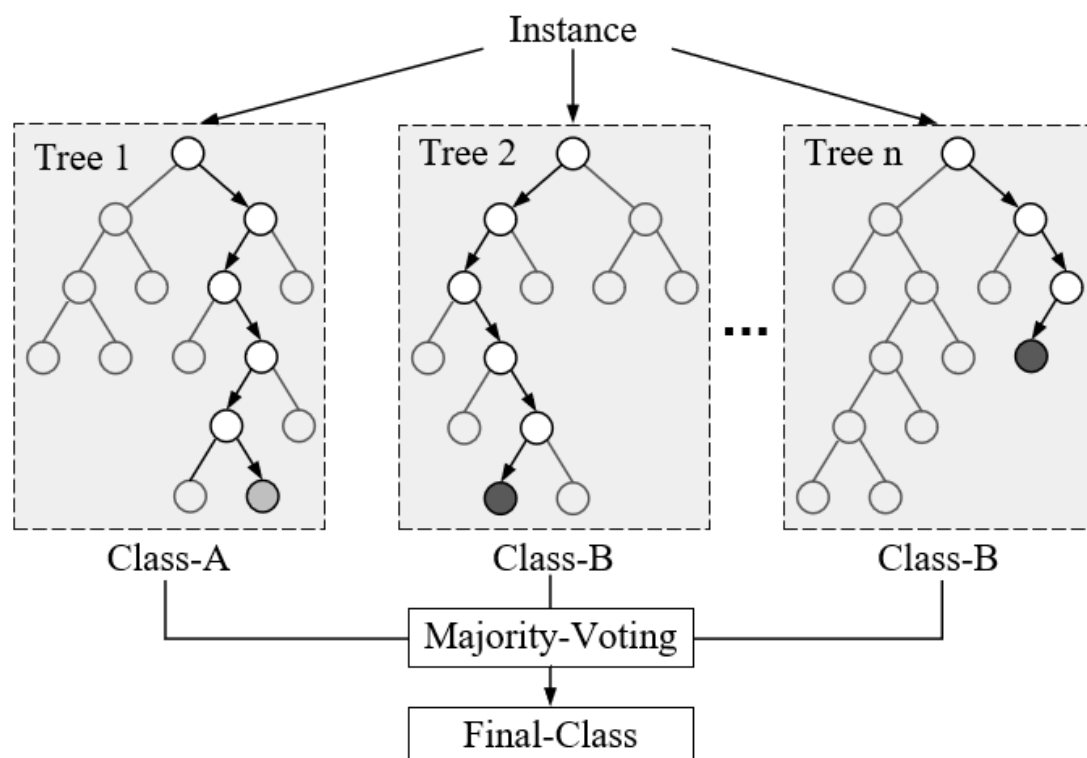


Figure 4-1. Random Forest

When used for classification, the prediction from the Random Forest is the result of a majority vote across all trees. For regression, it calculates the average of the outputs. This process repeats to build a forest of trees, and the collective decision is made based on weighted voting or averaging their individual predictions.

The strengths of the Random Forest algorithm are evident in its excellent performance with high-dimensional data and large datasets, and its robustness to

missing data. It can also identify the most influential features within the data, although this feature selection is inherently random and can be less accurate with smaller datasets. Another advantage is its resistance to overfitting during the training phase. Random Forest is versatile, finding utility in various domains like recommendation systems, credit scoring, and other areas that require classification or regression, making it a favored choice for practical application.

4.1.1 Optimizing a Random Forest Classifier with Grid Search

This study uses a Random Forest classifier using grid search to find the best combination of hyperparameters. It runs cross-validation on different 'n_estimators' and 'max_depth' values to maximize the F1 score for balanced class weights. After selecting the best parameters, the model is retrained on the training data. This approach aims to improve model accuracy while mitigating class imbalance issues.

```
# Initialize the random forest model with balanced class weights
rf = RandomForestClassifier(class_weight='balanced', random_state=42)

# Define the grid of parameters to search over
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
}

# Initialize the grid search
grid_search = GridSearchCV(rf, param_grid, cv=3, scoring='f1_macro', n_jobs=-1)

# Fit the grid search to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_

best_params

{'max_depth': 20, 'n_estimators': 200}

# Train the random forest model with the best parameters
rf_best = RandomForestClassifier(class_weight='balanced', random_state=42, **best_params)
rf_best.fit(X_train, y_train)

RandomForestClassifier(class_weight='balanced', max_depth=20, n_estimators=200,
                        random_state=42)
```

Figure 4-2. The code of Random Forest

4.1.2 Result of Random Forest

(1) The result of the random forest:

Accuracy of Model: 0.8875714285714286

Classification Report:

	precision	recall	f1-score	support
0	0.92	0.96	0.94	12125
1	0.61	0.43	0.51	1875
accuracy			0.89	14000
macro avg	0.77	0.69	0.72	14000
weighted avg	0.88	0.89	0.88	14000

Figure 4-3. Evaluation indicators of Random Forest

The model's accuracy is about 0.89, and it's more adept at predicting non-medalists (class '0') with a high F1-score of 0.94. However, its performance predicting medal winners (class '1') is lower, with an F1-score of 0.51, which may suggest a challenge with the less represented medal-winning class in the data.

(2) Confusion Matrix:

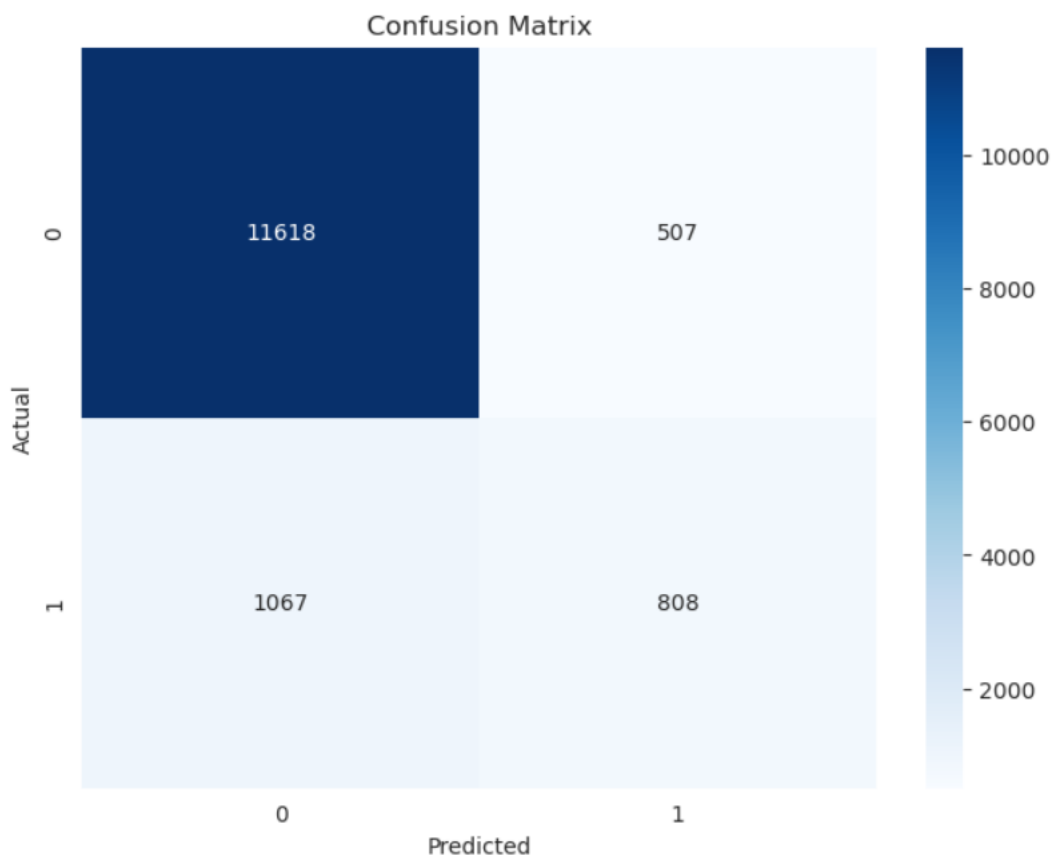


Figure 4-4. Confusion matrix of Random Forest

The confusion matrix shows that the model correctly predicted non-medalists (class '0') 11,618 times, but there were 507 false positives, where non-medalists were incorrectly labeled as medalists. For medal winners (class '1'), there were 808 true positives, where medalists were correctly identified, and 1,067 false negatives, where medalists were mistakenly labeled as non-medalists. This suggests that while the model is quite good at identifying athletes who didn't win medals, it has a relatively higher error rate when it comes to correctly identifying actual medal winners.

4.2 XGBoost (Python)

XGBoost is an advanced implementation of gradient boosting that uses decision trees in sequence to correct previous models' mistakes, optimizing both prediction accuracy and computing speed. It employs gradient descent to minimize error and includes regularization to prevent overfitting, making it a powerful and efficient tool for machine learning tasks.

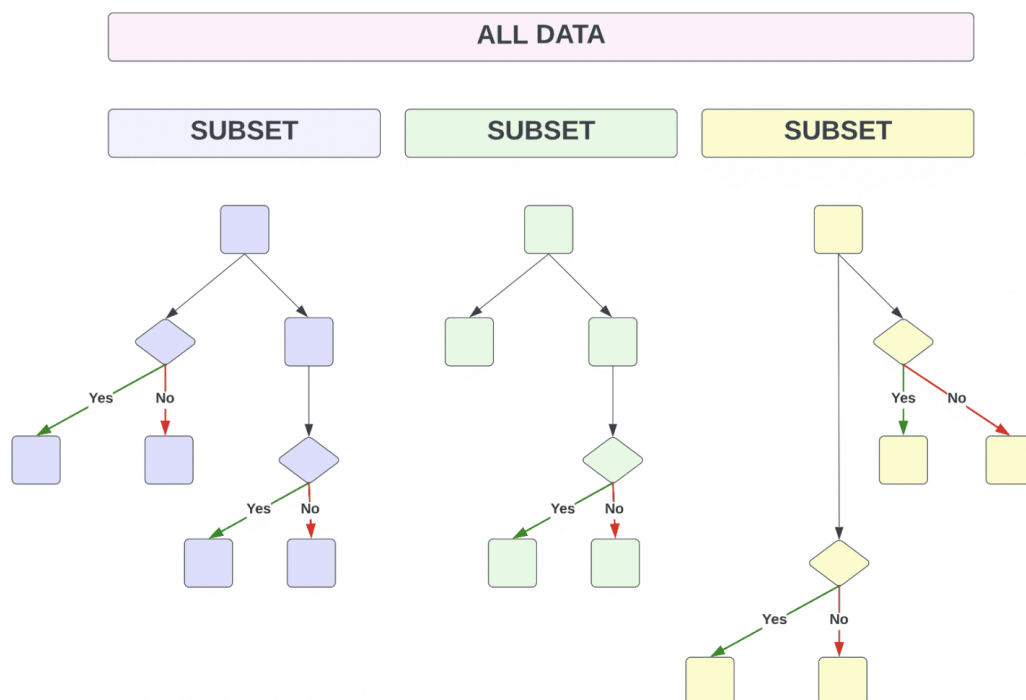


Figure 4-5. XGBoost

4.2.1 Optimizing a XGBoost Classifier with Grid Search

This study uses an XGBoost classifier using grid search. The XGBoost classifier is initialized with certain parameters. A parameter grid is defined with a range of values for 'n_estimators', 'max_depth', and 'learning_rate'. GridSearchCV is then used with 3-fold cross-validation and 'f1_macro' scoring to find the best parameters. The grid search is fitted to the training data, and the best parameters are extracted, showing an optimal learning rate of 0.2, a max depth of 30, and 200 estimators. These optimal parameters are then presumably used to train the XGBoost model.

```
from xgboost import XGBClassifier

# Initialize the XGBoost Classifier
xgb = XGBClassifier(random_state=42, use_label_encoder=False, objective='multi:softmax', num_class=2)

# Define the grid of parameters to search over
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [30, 40, 50],
    'learning_rate': [0.2, 0.3, 0.4],
}

# Initialize the grid search
grid_search = GridSearchCV(xgb, param_grid, cv=3, scoring='f1_macro', n_jobs=-1)

# Fit the grid search to the training data
grid_search.fit(X_train, y_train)

# Get the best parameters
best_params = grid_search.best_params_

best_params

{'learning_rate': 0.2, 'max_depth': 30, 'n_estimators': 200}
```

Figure 4-6. The code of XGBoost

4.2.2 Result of Random Forest

(1) The result of the random forest:

```
Accuracy of Model: 0.8994285714285715
Classification Report:
              precision    recall  f1-score   support

     0       0.92      0.97      0.94      12125
     1       0.69      0.46      0.55      1875

 accuracy          0.90      14000
 macro avg         0.80      0.71      0.75      14000
 weighted avg      0.89      0.90      0.89      14000
```

Figure 4-7. Evaluation indicators of XGBoost

The model exhibits an accuracy of approximately 0.90. It performs well in

identifying class '0' (likely non-medal winners), with high precision and recall leading to a strong F1-score of 0.94. However, it is less effective for class '1' (likely medal winners), with notably lower recall and an F1-score of 0.55. The overall weighted average scores indicate a better performance on the majority class, which has a significantly larger support. The macro average reflects the unweighted mean performance across classes.

(2) Confusion Matrix:

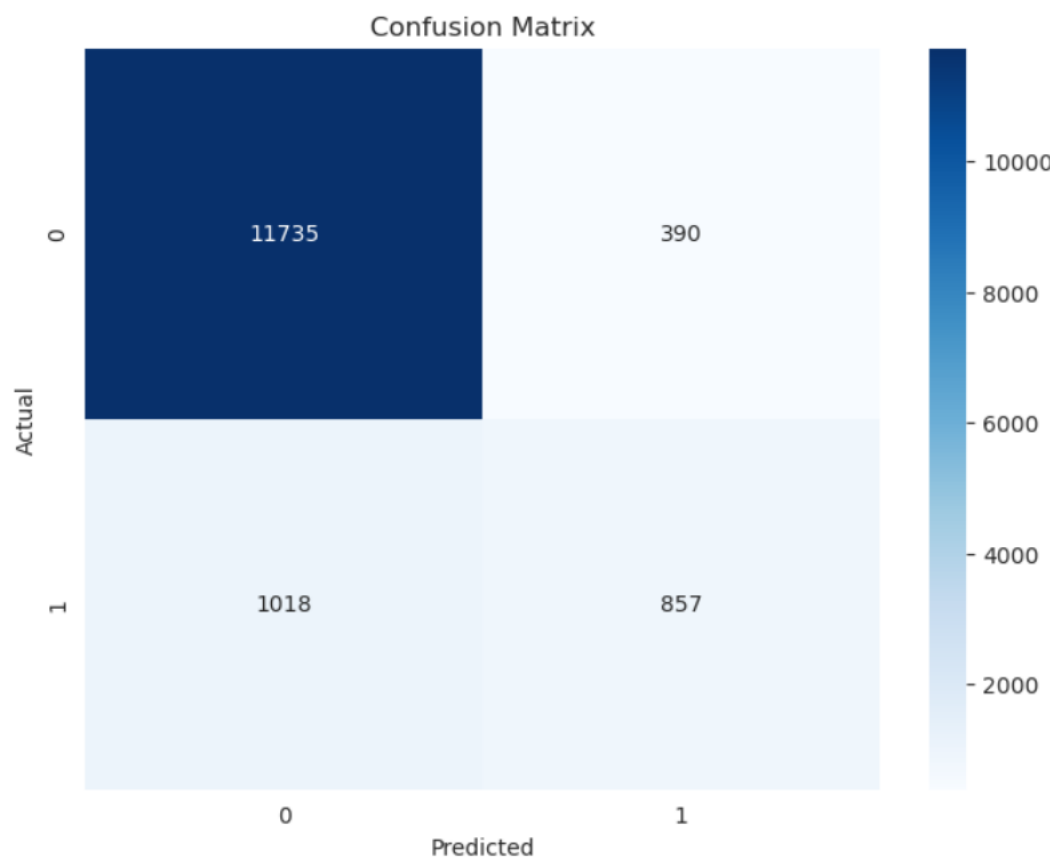


Figure 4-8. Confusion matrix of XGBoost

The confusion matrix shows the model's predictions versus the actual labels. It correctly predicted '0' (non-medal winners) 11,735 times and '1' (medal winners) 857 times. There were 390 instances where non-medal winners were incorrectly identified as medal winners (false positives), and 1,018 instances where medal winners were mistaken for non-medal winners (false negatives). This suggests the model is better at identifying non-medal winners than medal winners.

4.3 RandomTree (WEKA)

In Weka, the process for selecting random attributes and generating split points is as follows:

- ① A number of attributes, K , to be selected is set.
- ② Attributes are sampled without replacement from the entire set of attributes.
- ③ The information gain (not the gain ratio) of each attribute is calculated.
- ④ This process is repeated K times, and the attribute with the highest information gain is chosen as the split node.
- ⑤ Child subtrees for this node are then constructed.

4.3.1 Load the data and use the RandomTree classifier

- (1) load the csv file and turn it into *.arff

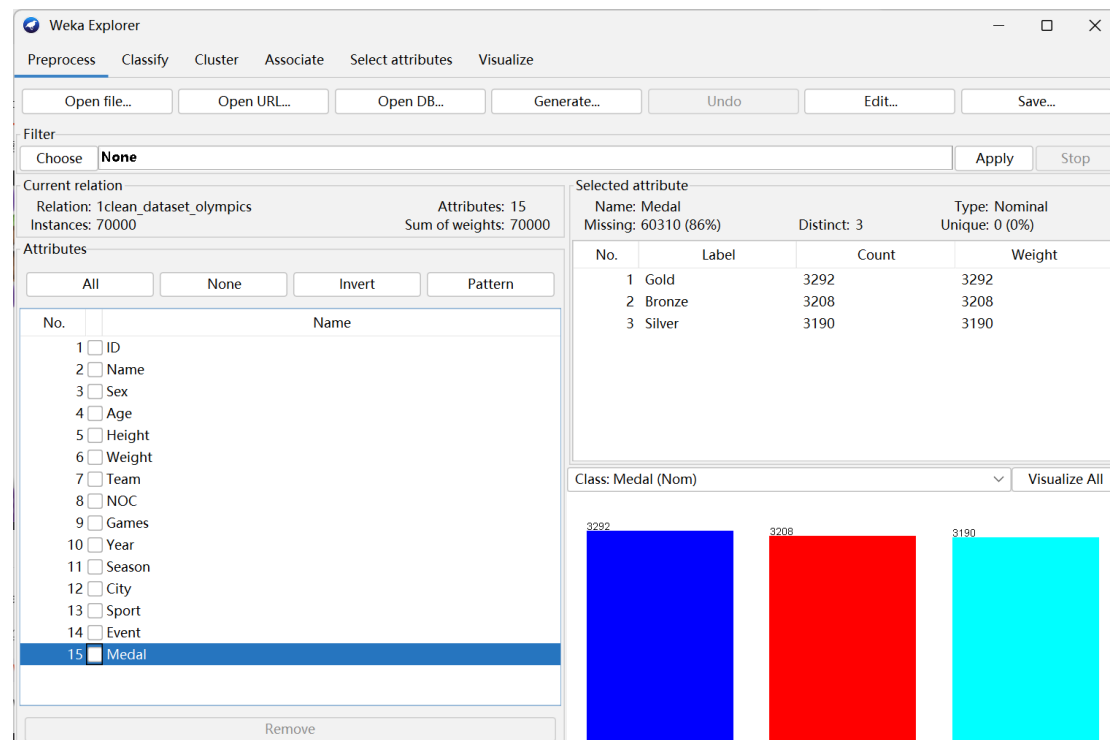


Figure 4-9. Load data to WEKA

- (2) replace the missing value in age, height, and weight and create a binary medal, 1 if the athlete won a medal, 0 otherwise

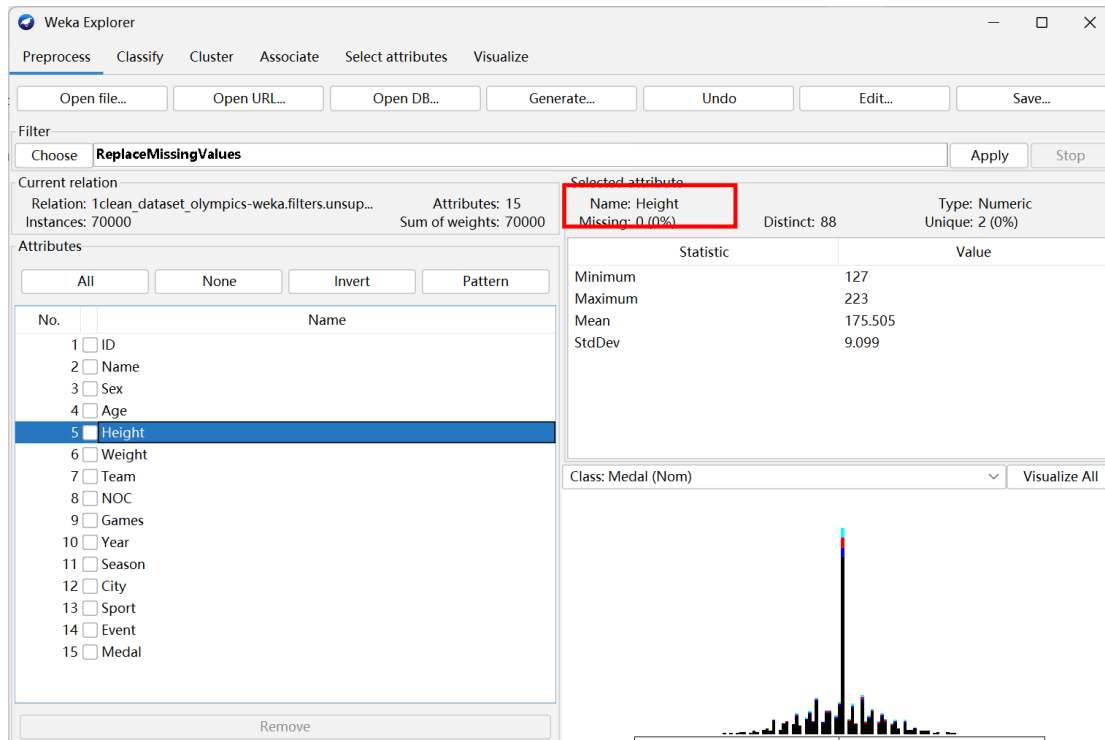


Figure 4-10. Replace the miss value

(3) filter the id, name, and year and show the remaining attributes

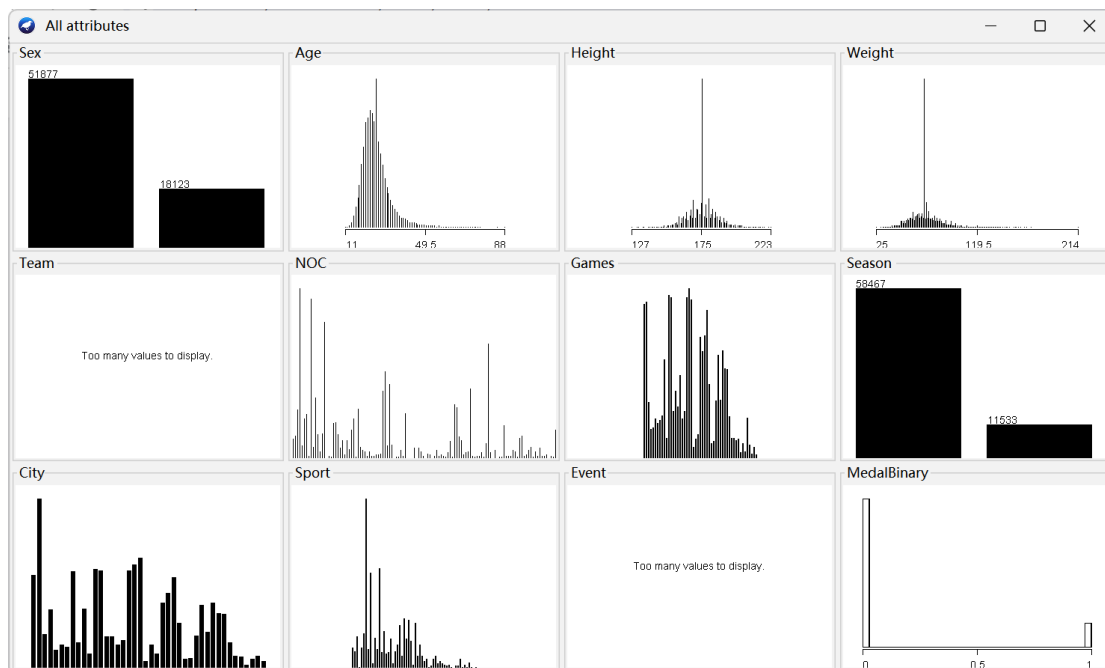


Figure 4-11. Show the remaining attributes

(4) use the RandomTree classifier to predict the medal

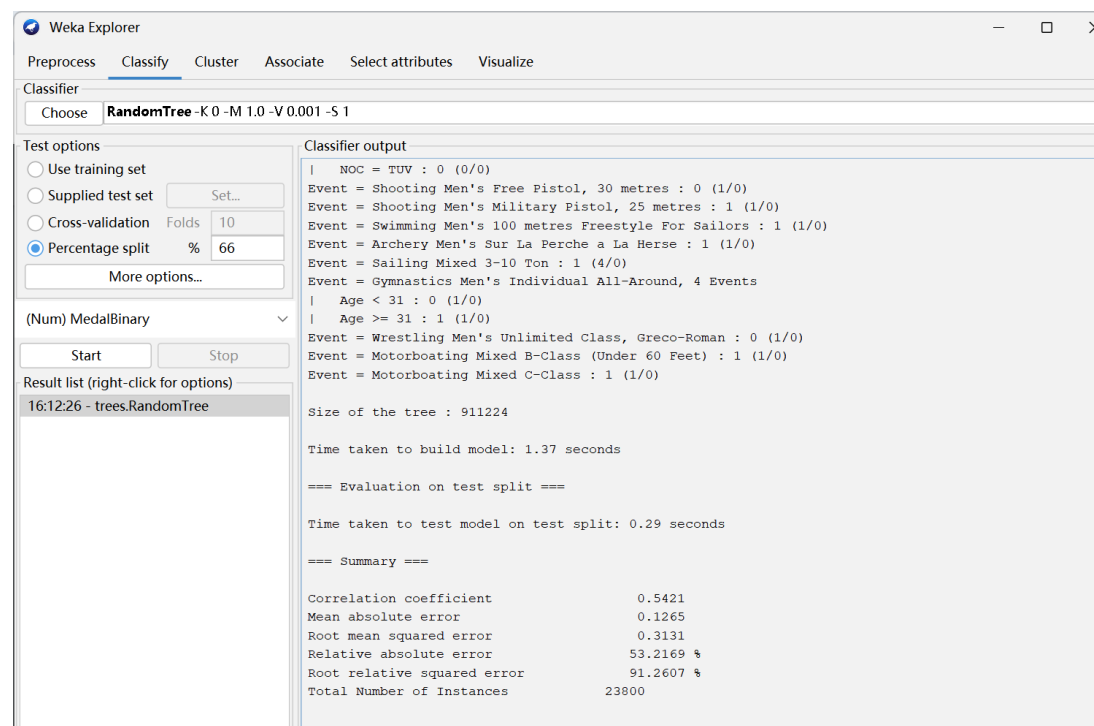


Figure 4-12. Use the RandomTree classifier to predict the medal

The performance metrics indicate a moderate correlation coefficient of 0.5421, suggesting that the model has a reasonable ability to predict outcomes linearly related to the true values. Additionally, the mean absolute error and root mean squared error are 0.1265 and 0.3131, respectively, indicating that the model predictions are fairly accurate but there is still some room for improvement in terms of reducing prediction errors.

Question2—How has the number of participating athletes from each country changed over approximately 100 years, from the 1910s to the 2010s? (Tableau)

This analysis can show the changes in engagement and interest in sports by country. It also may reflect the development of sports infrastructure, investment, and athlete development policies in each country and can demonstrate how certain countries have grown into sports powerhouses over time. It groups data by season and year, categorizes years into decades, merges the datasets for better comparison, and employs world maps that depict these changes using color gradients to indicate the number of athletes from each country. This visualization aims to provide a clear,

visual representation of trends and shifts in athlete participation across the Olympics over the decades.

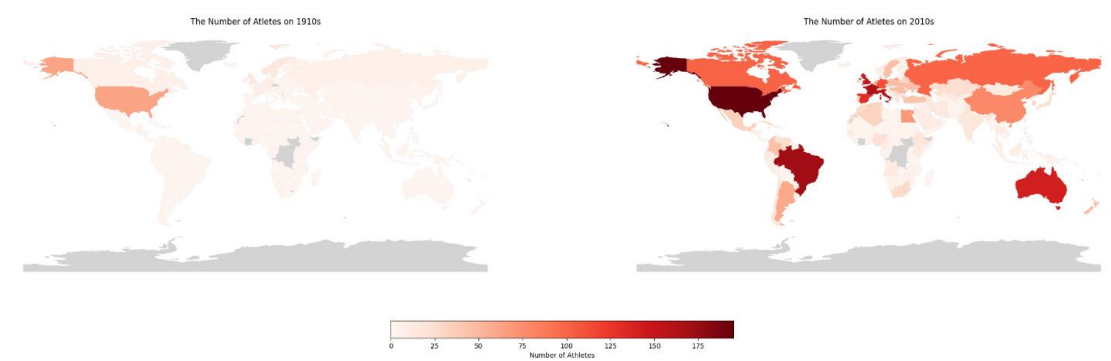


Figure 4-13. Summer Olympic

There is a significant increase in athlete participation evident in countries such as the USA, China, and Russia from the 1910s to the 2010s. This likely reflects enhancements in sports infrastructure, athlete training, and Olympic investment in these regions.

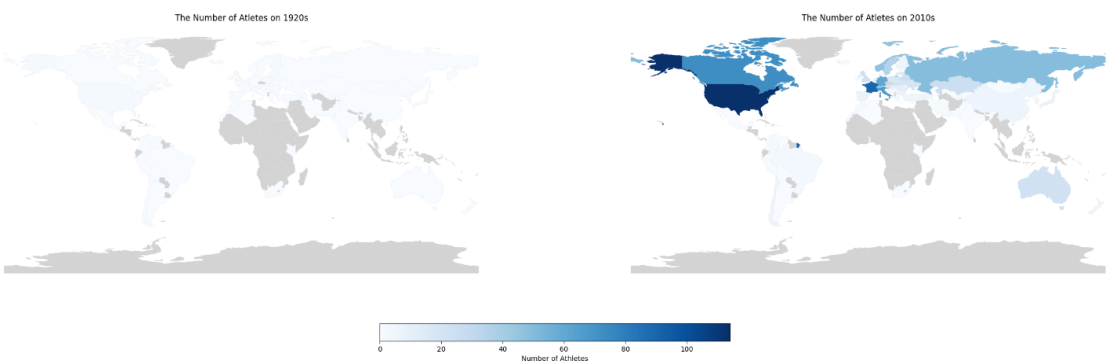


Figure 4-14. Winter Olympic

From the 1910s to the 2010s, many countries in North America and Europe show an increase in the number of athletes, particularly Canada and Russia, where the deep blue color highlights a significant rise in participation.

Question3——How has the number of events in the Winter and Summer Olympics changed over the years? (Python)

The evolution of diversity and inclusivity in the Olympics and can reflect the sports trends of specific eras. It defines a function `get_event_diff` that calculates the yearly difference in event counts and the rate of change from one year to the next. This data is then visualized using the `visualize_event_diff` function, which plots these changes using dual axes: one for the absolute change in events and another for the rate of change, enhancing the understanding of trends in Olympic event evolution. Different colors are used for each Olympic season to distinguish between the Summer and Winter Games, highlighting the dynamics of event changes over the years.

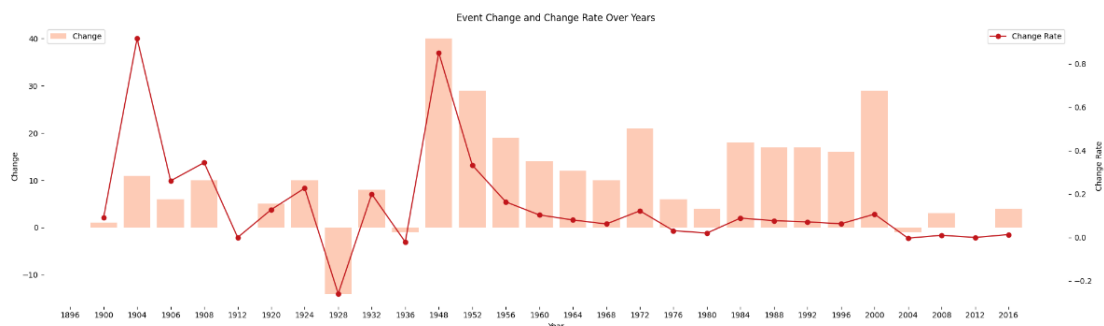


Figure 4-15. Summer Olympic

The chart shows significant fluctuations in the number of events through the years, with a notable spike in the early 20th century followed by a sharp decline. This could represent the stabilization of event categories or major global disruptions such as wars.

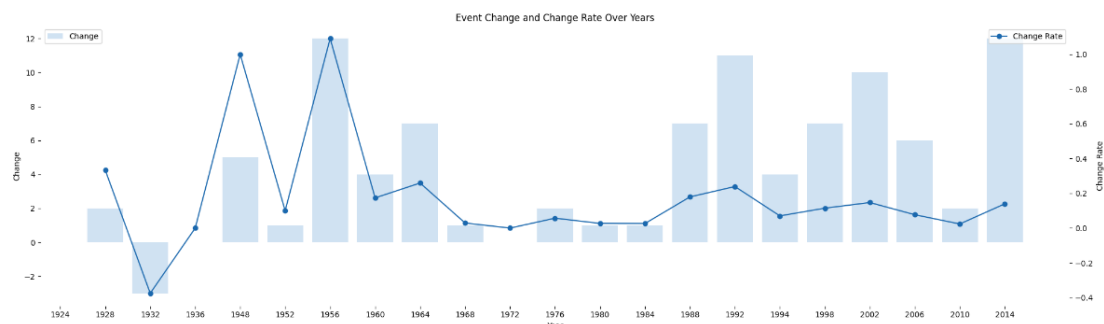


Figure 4-16. Winter Olympic

The rate of change occasionally spikes, suggesting specific years where significant changes in the number of events occurred. This might be due to the introduction of new winter sports or modifications in the sports categories.

Questio4—How does the proportion of medal colors vary among the top 30 countries with the most medals? (Python)

It calculates the total medals for each sport per country, identifying the top performers, and then converts these counts into percentage values for a normalized comparison. The results are displayed in heatmaps, providing a clear visual representation of how countries perform across different sports in both Olympic seasons.

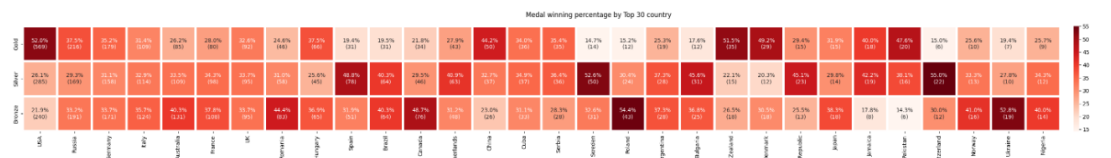


Figure 4-17. Summer Olympic

Countries like the USA and China show a broader range of reds across many sports, indicating a widespread ability to win medals in various disciplines. Some countries, like Jamaica in athletics and Hungary in water sports (such as swimming and water polo), demonstrate specialization, shown by the intense red colors in those specific sports.

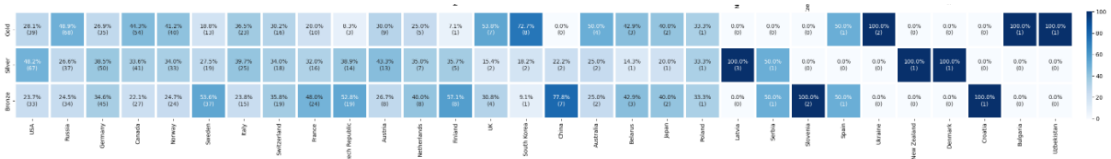


Figure 4-18. Winter Olympic

There is a higher concentration of intense colors in fewer sports, reflecting the more specialized nature of winter sports competition. Countries such as Norway and Canada show deep blue colors in skiing and ice hockey, respectively, indicating their strength and historical success in these sports.

5 Challenge and Reflection

5.1 Challenge

(1) Problem of loading data:

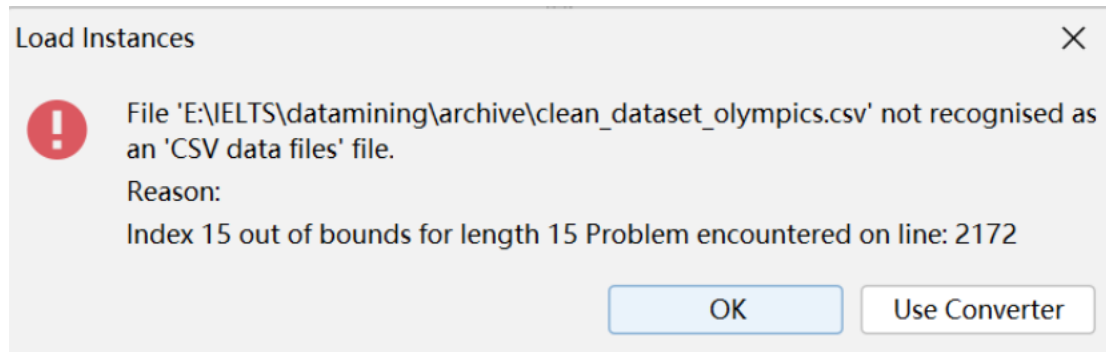


Figure 5-1. Problem of loading data

Solution: The issue arose because there are special characters in the 'name' and 'team' fields. To resolve this, I need to preprocess the file with Python to handle these special characters before importing it.

5.2 Learned

(1) Data Preparation and Cleaning: Using Weka has improved my understanding of preparing and cleaning data, especially in dealing with special characters and formatting issues that can cause errors during data import. Learning to preprocess data effectively to ensure it's suitable for analysis is a crucial skill I developed.

(2) Visualizations: Creating impactful visualizations in Tableau has been an essential part of my learning. The ability to visually represent complex datasets has provided practical experience in designing and interpreting various types of visual data presentations, such as heatmaps and trend analyses, which are instrumental in conveying insights clearly and effectively.

(3) Machine Learning and Data Mining Techniques: Weka has introduced me to practical applications of machine learning algorithms and data mining techniques. Understanding how to apply classifiers, cluster analysis, and association rules in Weka has reinforced the importance of these methods in extracting patterns and making predictions based on large datasets.

5.3 Reflection and Future

In future projects, I intend to explore and analyze relationships between a wider range of variables within the datasets. Using Weka's attribute selection and evaluation modules, I can identify which variables have the most significant impact on outcomes and should be included in the models. This will help in building more accurate and reliable predictive models.

6 Conclusion

The analysis of Olympic Games data using Python, Tableau, and Weka helps us understand the complexities and dynamics of the Games. This study manipulated and prepared data, enabling detailed statistical analysis and the application of machine learning models by Python and WEKA. Tableau was instrumental in visualizing the data, providing clear and impactful insights into trends and patterns across different Olympic sports and countries. This integrated approach highlights the importance of using diverse analytical tools to tackle the complexity of Olympic data, ensuring a thorough exploration of the factors that contribute to the success and appeal of the Games.