OLYMPICS DATA ANALYSIS WITH PREDICTION

MINI PROJECT

VARSHA U
ROLL NO:28
S3 MCA

GUIDED BY:
PROF.MANU JOHN

INTRODUCTION

- Objective: Analyzing historical Olympic data using machine learning to predict future outcomes.
- Focus Areas: Athlete performance, country performance, event trends, and medal predictions.
- Benefits: Provides insights, identifies patterns, and supports data-driven decision-making.
- Specialization: Falls within the domain of specialized sports analytics.

REFERENCE PAPERS

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 was to evaluate three boosting algorithms (LGBM, XGBoost, CatBoost) for predicting Olympic medal outcomes.

- Accuracy :
  - CatBoost: 89.1%
  - LightGBM: 90.1%
  - XGBoost: 90.2%

- The dataset covered 11 Olympic Games and included various athlete attributes like nationality, sport, age, and historical performance, making it comprehensive for analysis.

- XGBoost emerged as the leading algorithm for precise Olympic medal predictions, showcasing its potential in sports analytics.

REFERENCE PAPERS

pape 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.

- The research aimed to predict Summer Olympic event winners using a Random Forest Regression model.
- accuracy achieved : 89.76%
- The dataset covered athletes from 1896 to 2016 Olympics and integrated external data on world population and GDP for enhanced predictions.
- This study contributed valuable insights into Olympic success factors and demonstrated the practical use of machine learning in sports analytics for outcome predictions.

REFERENCE PAPERS

paper 3:

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

- Aimed to forecast Olympic medal counts for various nations in Tokyo 2020 using a two-staged Random Forest model, trained on a dataset of socio-economic variables from 1991 to 2020.

- accuracy : 85.38%.

- The dataset contains GDP, population, athlete count, COVID-19 impact, host country status, political regime, and geographic region data from various organizations.

- The model outperformed other machine learning algorithms and achieved an impressive accuracy rate, offering valuable insights into the predictive power of socio-economic factors on Olympic medal counts.

Insights from three studies

	Title	Year	Publisher	Summary
1	A Comparative Study of Different Boosting Algorithms for Predicting Olympic Medal	2022	IEEE	Algorithm selected: XGBoost Accuracy: 90% Dataset : Records of Olympics history from the earliest competition in 1896 to recent games in 2016.
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.
3	Forecasting the Olympic medal distribution–a socioeconomic machine learning model	2022	Elsevier	Algorithm : Random Forest accuracy : 85.38%. Dataset: GDP, population, the number of athletes, the impact of COVID-19, host country status, political regime, and geographic region.

PROJECT:

- Project Title: Olympics data analysis with Prediction -

- It's a Exploratory Data Analysis of the Modern Olympic Games and provides valuable insights and predictions that can be used for data-driven strategies for future Olympic success

Methodology:

- Data collection and cleaning
- Exploratory data analysis
- Visualization
- Modeling

Algorithm:

- Comparative study of
- Random forest
- XGBoost

Expected Results:

- Medal tally
- Overall analysis
- Country-wise analysis
- Athlete-wise analysis
- medal prediction

DATASET EXPLORATION

The dataset is collected from kaggle. The dataset contains two files: athlete_events.csv and noc_regions.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 centimetres
- 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

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)
- region
- notes

Joining the NOC data

```
noc.drop('notes', axis=1, inplace=True)
df = df.merge(right=noc, on='NOC', how='left')
```

ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal	region
1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN	China
2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN	China
3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN	Denmark
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	Denmark
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	Netherlands

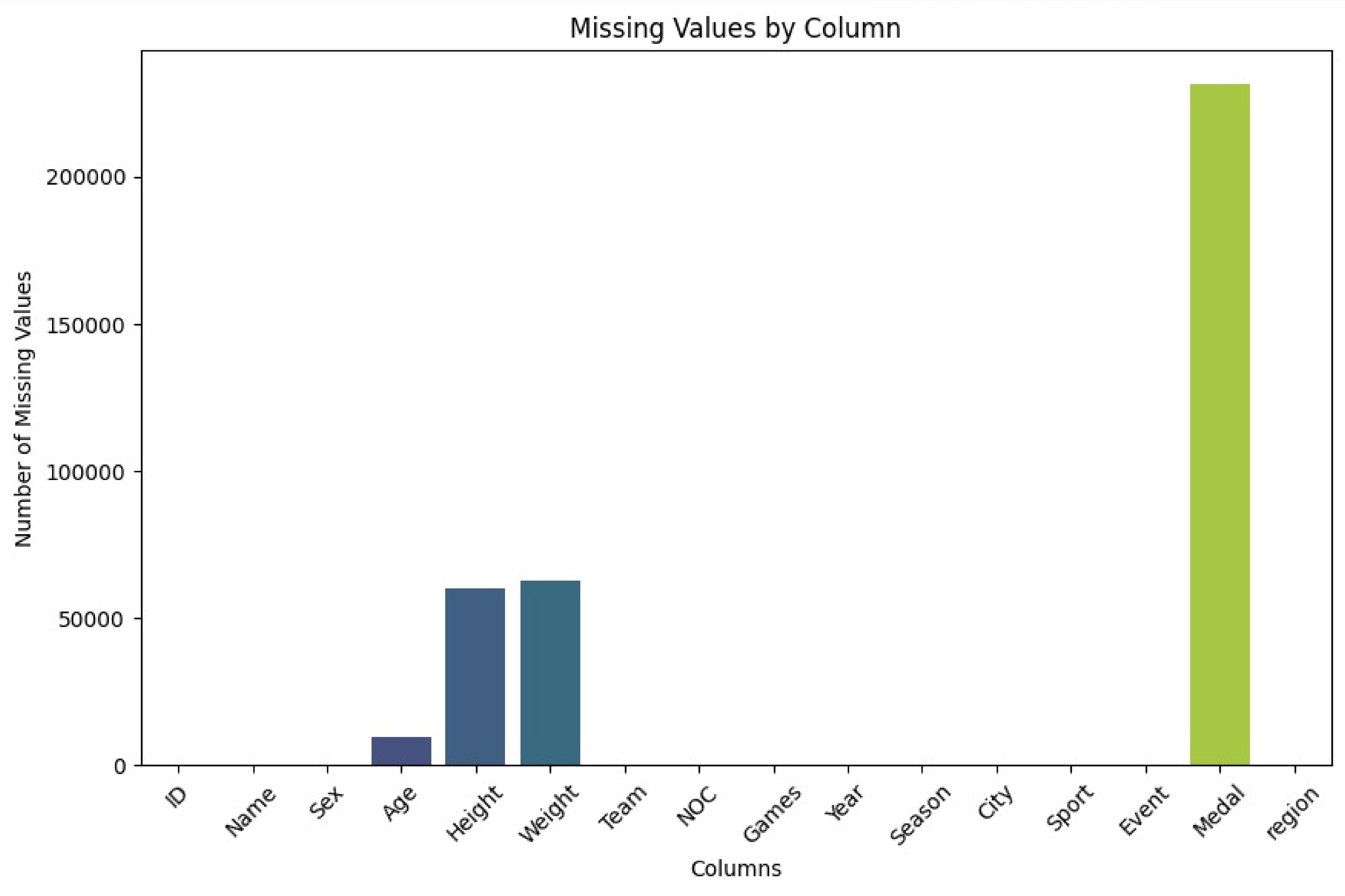
```
Number of Unique Values in Each Column:
ID        135571
Name      134732
Sex            2
Age           74
Height        95
Weight       220
Team        1184
NOC          230
Games         51
Year          35
Season         2
City          42
Sport         66
Event        765
Medal          3
region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

Missing Values

Checking Nulls

```
# Check for Nulls .
df.isna().sum()
```

```
ID              0
Name            0
Sex             0
Age          9474
Height      60171
Weight      62875
Team            0
NOC             0
Games           0
Year            0
Season          0
City            0
Sport           0
Event           0
Medal      231333
region        370
dtype: int64
```



Treating missing values

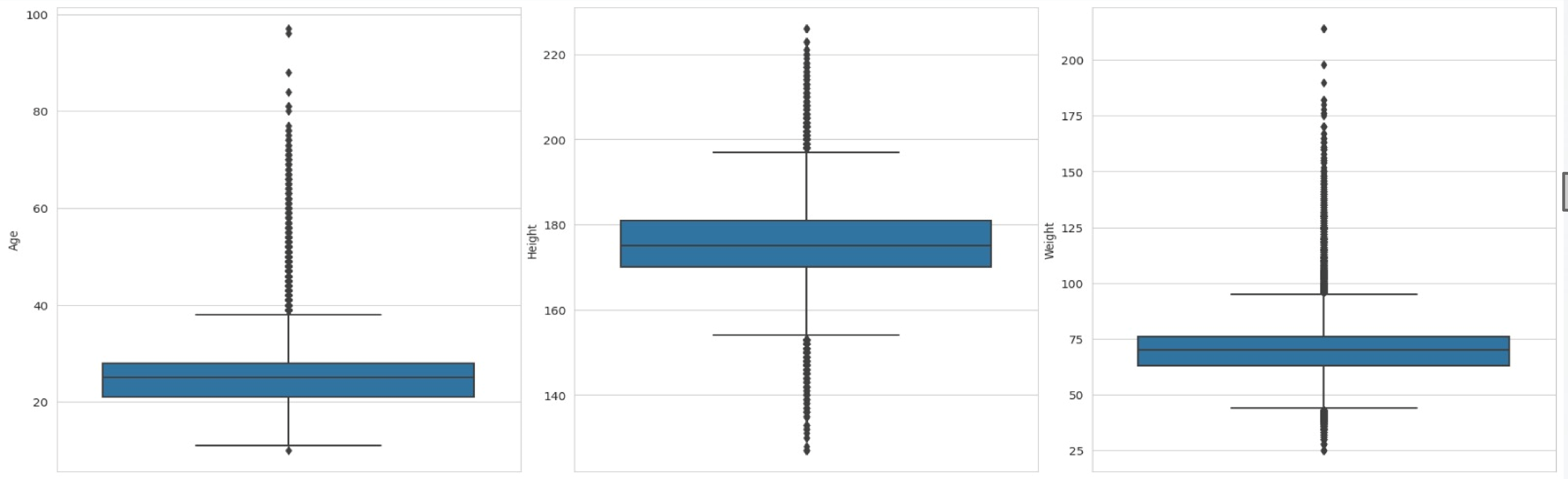
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 271116 entries, 0 to 271115
Data columns (total 16 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      271116 non-null  int64
 1   Name    271116 non-null  object
 2   Sex     271116 non-null  object
 3   Age     261642 non-null  float64
 4   Height  210945 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object
 7   NOC     271116 non-null  object
 8   Games   271116 non-null  object
 9   Year    271116 non-null  int64
 10  Season  271116 non-null  object
 11  City    271116 non-null  object
 12  Sport   271116 non-null  object
 13  Event   271116 non-null  object
 14  Medal   39783 non-null   object
 15  region  270746 non-null  object
dtypes: float64(3), int64(2), object(11)
memory usage: 35.2+ MB
```

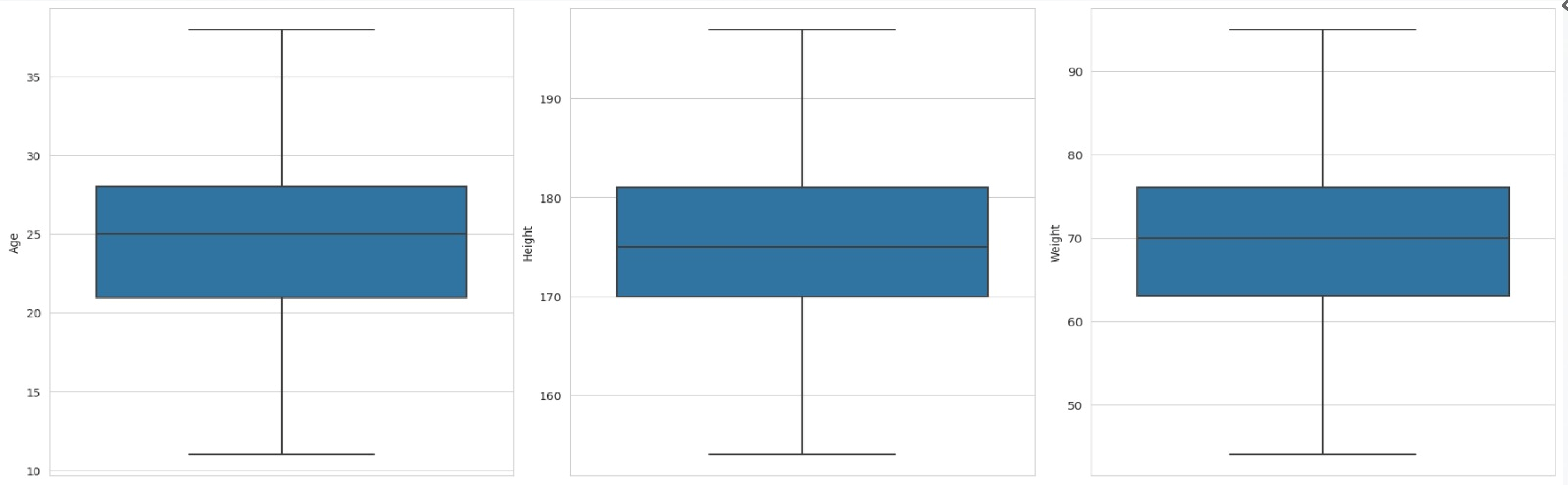
- Missing values in each numeric column are replaced with the mean value of that column.

- Missing values in each categorical column are replaced with the string 'None'.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262156 entries, 0 to 262155
Data columns (total 16 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   ID      262156 non-null  int64
 1   Name    262156 non-null  object
 2   Age     262156 non-null  int64
 3   Sex     262156 non-null  object
 4   Height  262156 non-null  int64
 5   Weight  262156 non-null  int64
 6   Year    262156 non-null  int64
 7   Team    262156 non-null  object
 8   NOC     262156 non-null  object
 9   region  262156 non-null  object
 10  Games   262156 non-null  object
 11  Season  262156 non-null  object
 12  City    262156 non-null  object
 13  Sport   262156 non-null  object
 14  Event   262156 non-null  object
 15  Medal   262156 non-null  object
dtypes: int64(5), object(11)
memory usage: 32.0+ MB
```

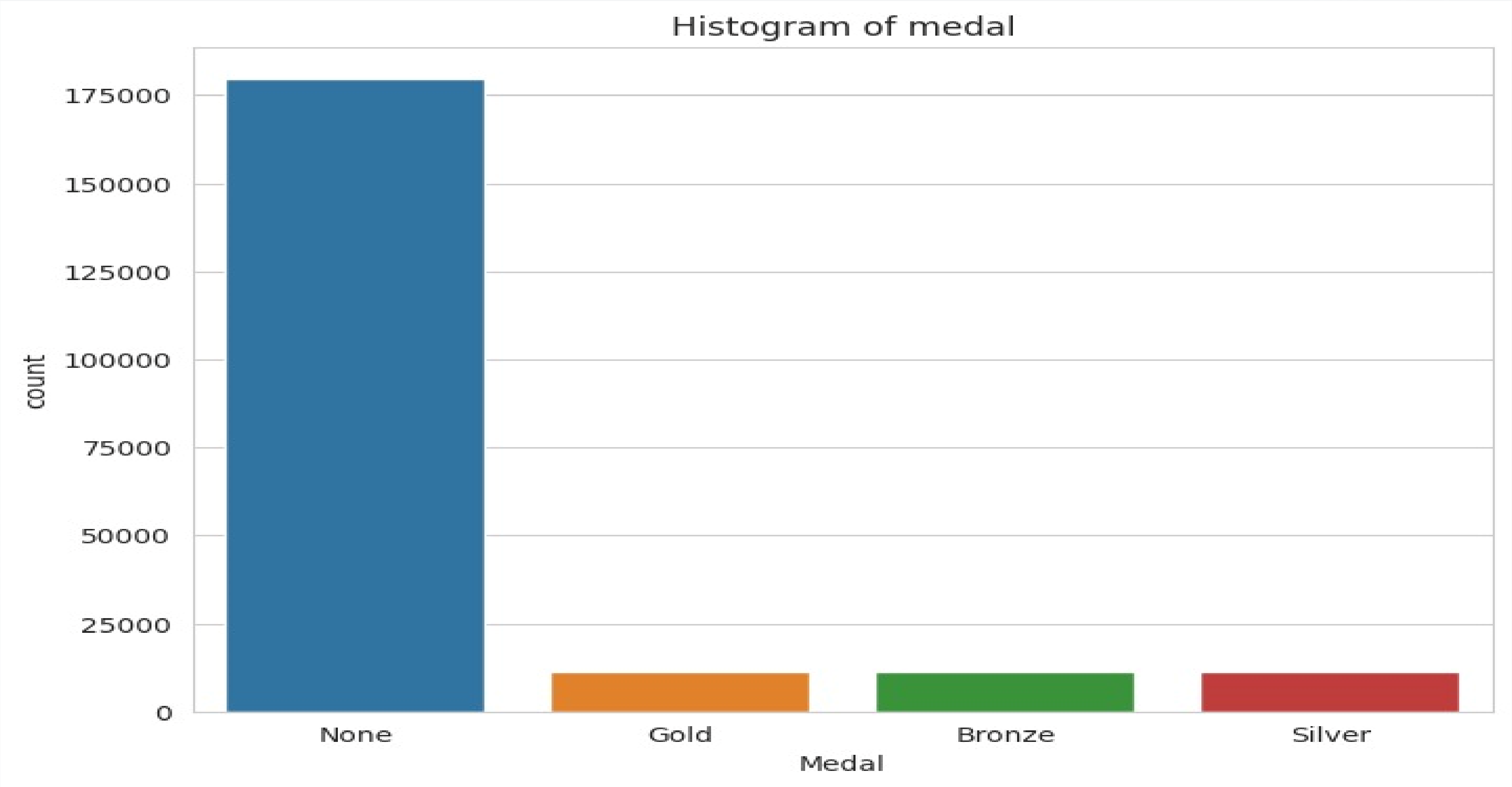
Box plot to show Outliers





Outlier Removal Using IQR Method for 'Weight' Column in Pandas DataFrame.

Checking consistency of class variable



Balancing Class Distribution Using RandomOverSampler in imbalanced-learn.

```
# 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
```

Selected features for medal prediction

data = data[['Age', 'Sex', 'Height', 'Weight', 'region', 'City', 'Sport', 'Medal’]]

Data type: 'Age', 'Height', 'Weight' are numerical & 'Sex', 'region', 'City', 'Sport’ are categorical.

Feature variables: 'Age', 'Sex', 'Height', 'Weight', 'region', 'City', 'Sport' are features.

Class variables: 'Medal' is the class variable.

Class labels: Gold ,Silver, Bronze ,None

Study of algorithms

1. Extreme Gradient Boosting
2. Random forest classifier

XGBoost

- eXtreme Gradient Boosted trees
- Remember boosting is an ensemble method
  - Each tree boosts attributes that led to mis-classifications of previous tree
- It is AMAZING
  - Routinely wins Kaggle competitions
  - Easy to use
  - Fast
  - A good choice for an algorithm to start with

Gradient 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

Features of XGBoost

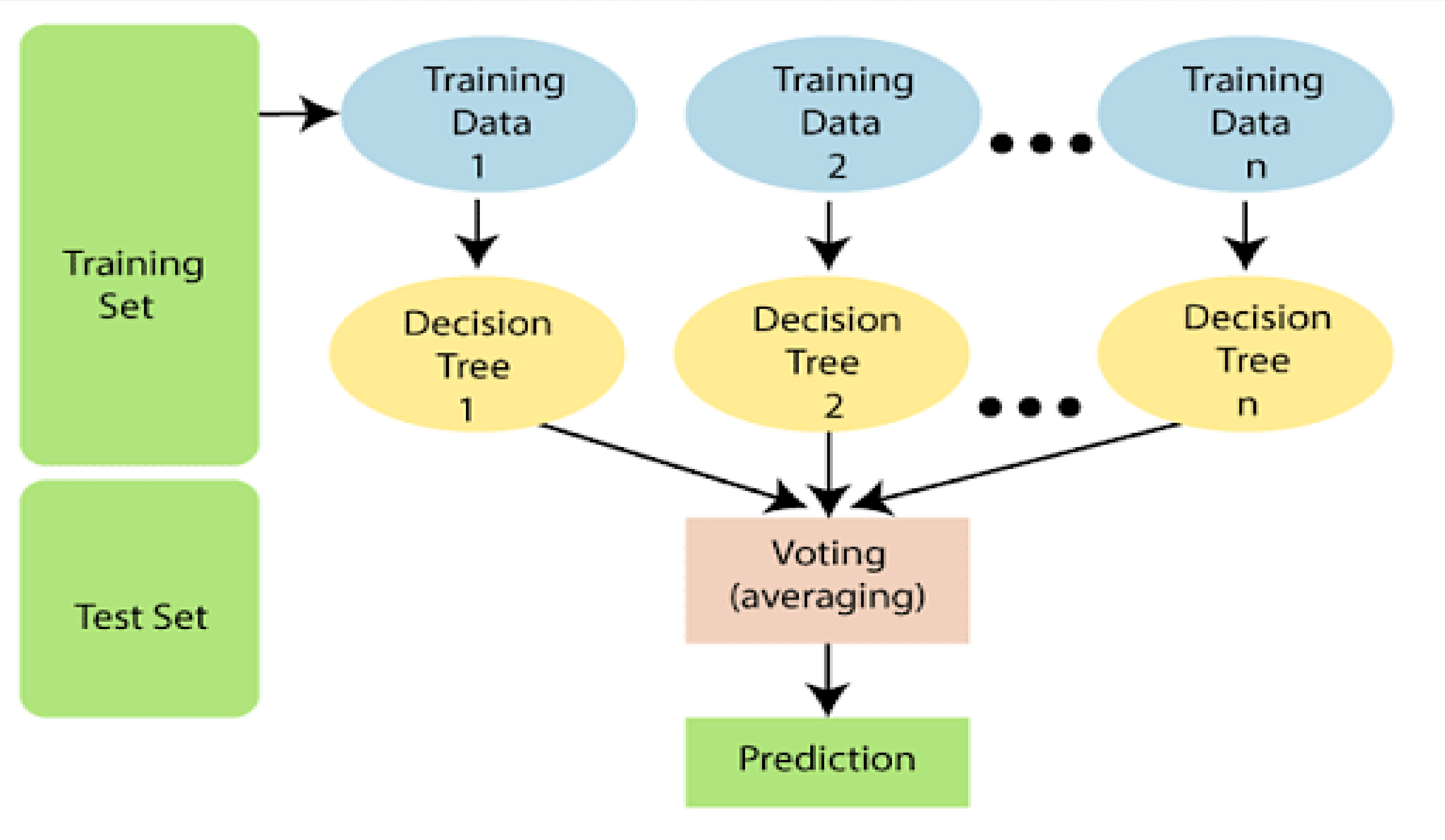
- Regularized boosting (prevents overfitting)
- Can handle missing values automatically
- Parallel processing
- Can cross-validate at each iteration
  - Enables early stopping, finding optimal number of iterations
- Incremental training
- Can plug in your own optimization objectives
- Tree pruning
  - Generally results in deeper, but optimized, trees

Random Forest Algorithm

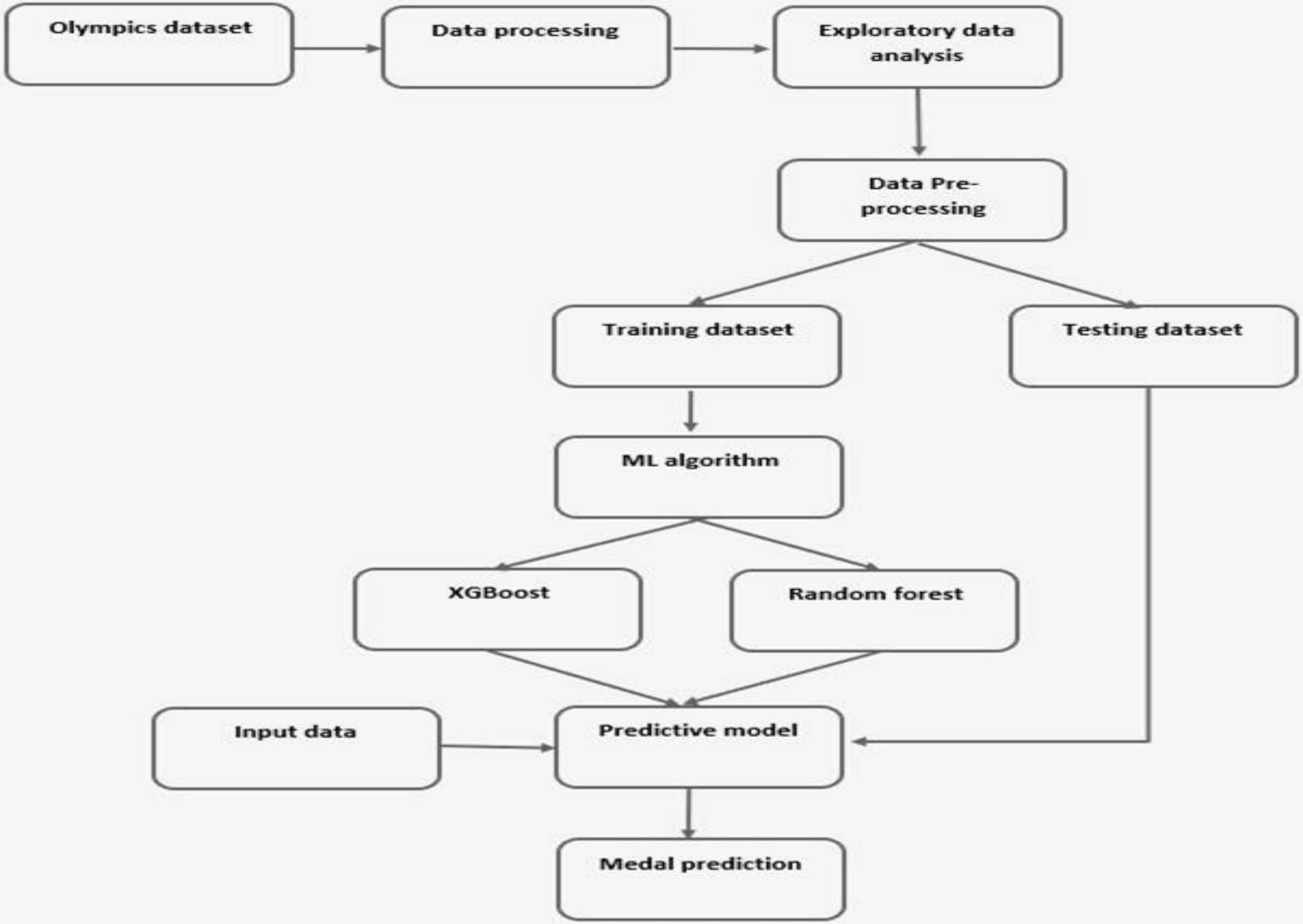
- Random forest is a commonly-used machine learning algorithm.
- A random forest is an ensemble learning method where multiple decision trees are constructed and then they are merged to get a more accurate prediction.
- Random forest became popular because of its ease of use and flexibility in handling both classification and regression problems.

How random forest works

1. Bootstrap Sampling
2. Decision Tree Building
3. Voting or Averageing



Project pipeline



Data splitting

```
# 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)
```

Training using Random forest

```
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)

tr = str(random_forest_olympics.score(X_train, y_train) * 100)
print("Training Score:", tr)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `spars
  warnings.warn(
Training Score: 93.03233299314465
```

Training using XGBOOST

```
# 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
)
# Build the model
xgboost_olympics = make_pipeline(full_pipe, xgboost_model)

# Train the model
xgboost_olympics.fit(X_train, y_train)
tr1 = str(xgboost_olympics.score(X_train, y_train) * 100)
print("Training Score:", tr1)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `sparse` was renam
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [12:01:41] WARNING: /workspace/src/lear
Parameters: { "scale_pos_weight" } are not used.

  warnings.warn(smsg, UserWarning)
Training Score: 93.03233299314465
```

Prediction Using Testing Data

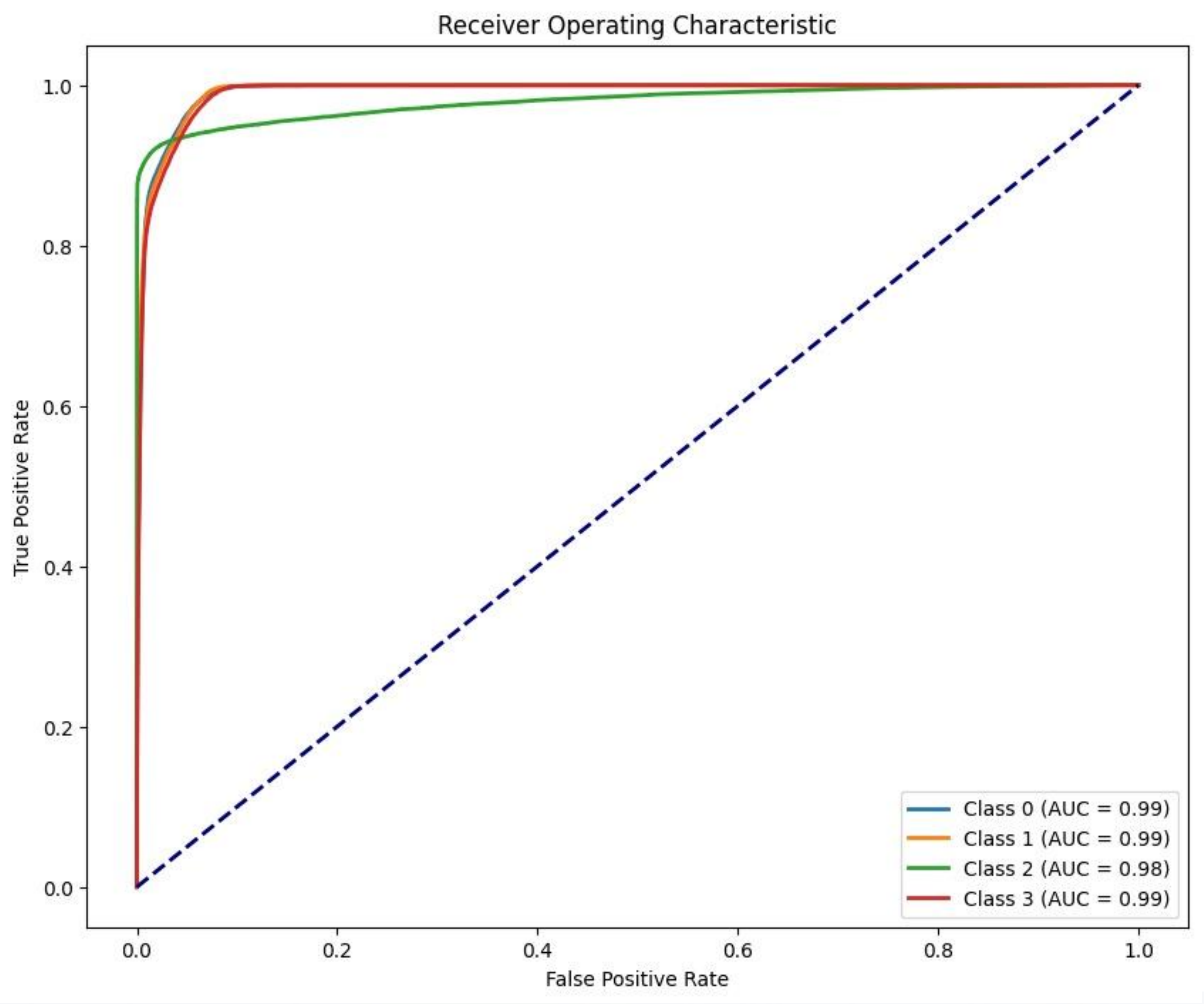
```python
# Make predictions on the test set
y_pred = random_forest_olympics.predict(X_test)

# 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}%')
```

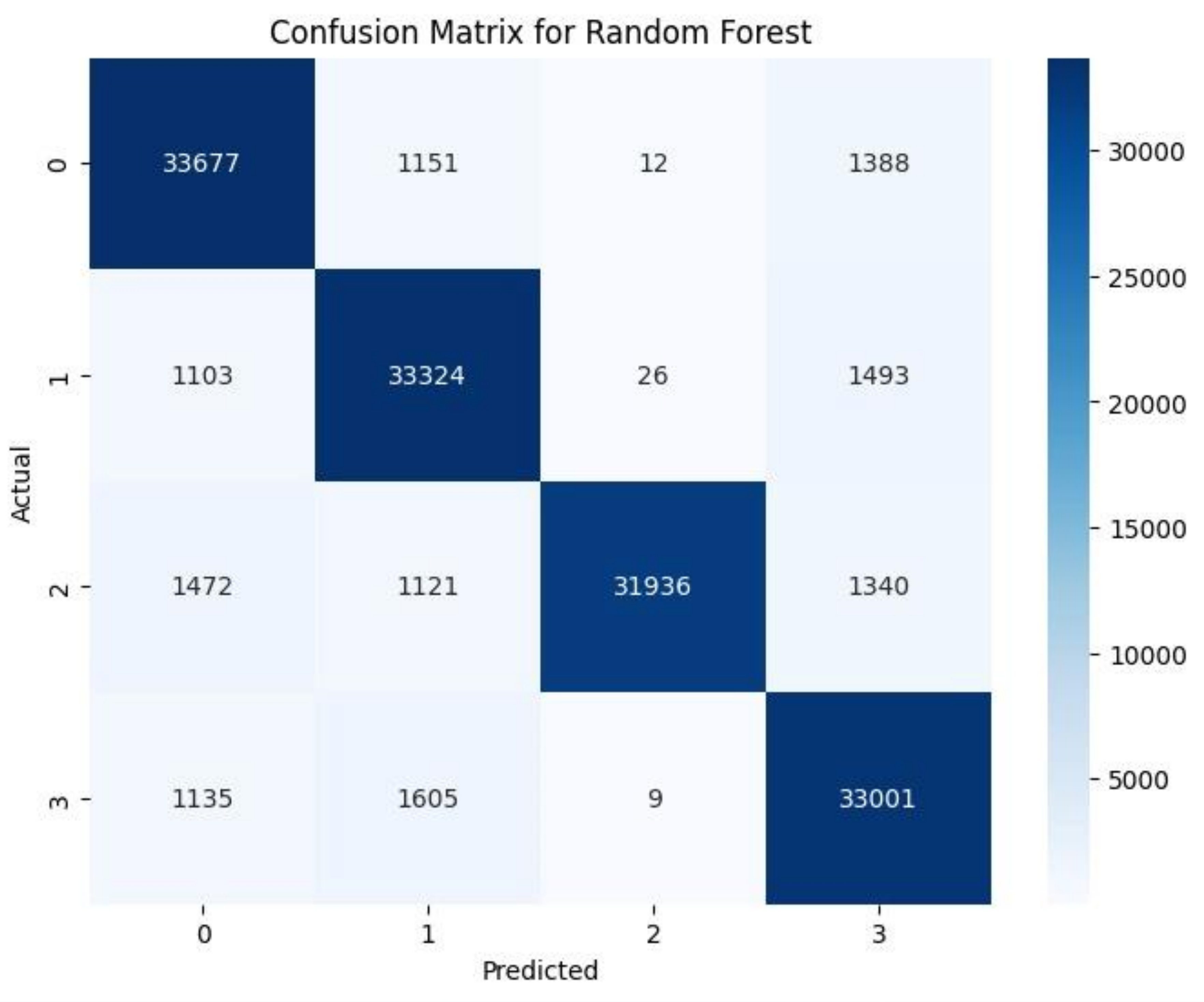
```
Random Forest Accuracy: 91.8%
Random Forest Precision: 92.0%
Random Forest Recall: 91.8%
```

```python
# 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}%')
```

```
Accuracy: 91.1%
Precision: 91.5%
Recall: 91.1%
```

Random Forest Has high accuracy, so we select random forest for building Model





User interface

localhost:8501

Deploy

Olympics Data Analysis

Select an Option

- Medal Tally
- Overall Analysis
- Country-wise Analysis
- Athlete-wise Analysis
- Medal Prediction

Medal Tally

Select Year

Overall

Select Country

Overall

Overall Tally

	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

localhost:8501

Deploy

Olympics Data Analysis

Select an Option

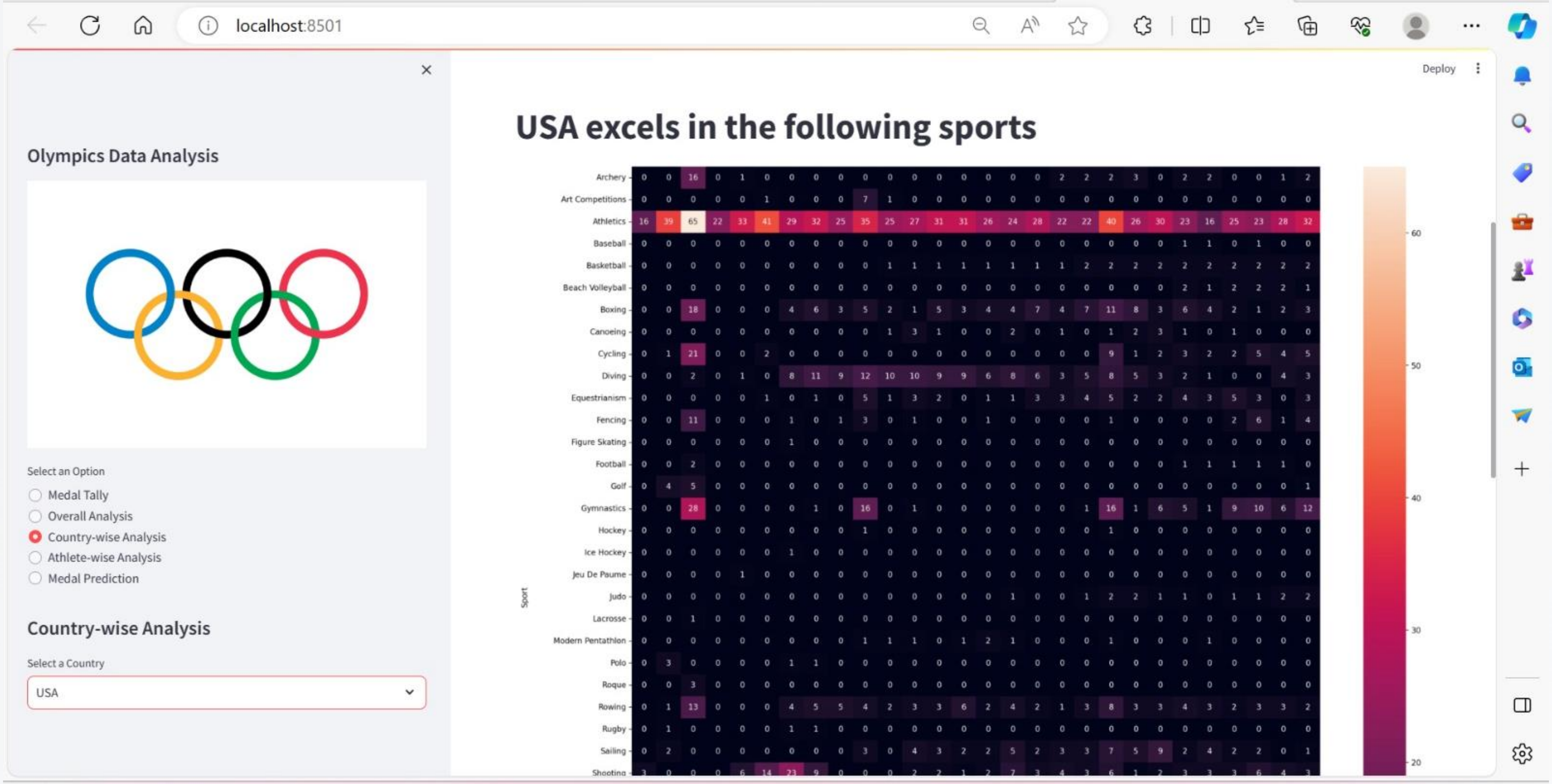
- Medal Tally
- Overall Analysis
- Country-wise Analysis
- Athlete-wise Analysis
- Medal Prediction

Top Statistics

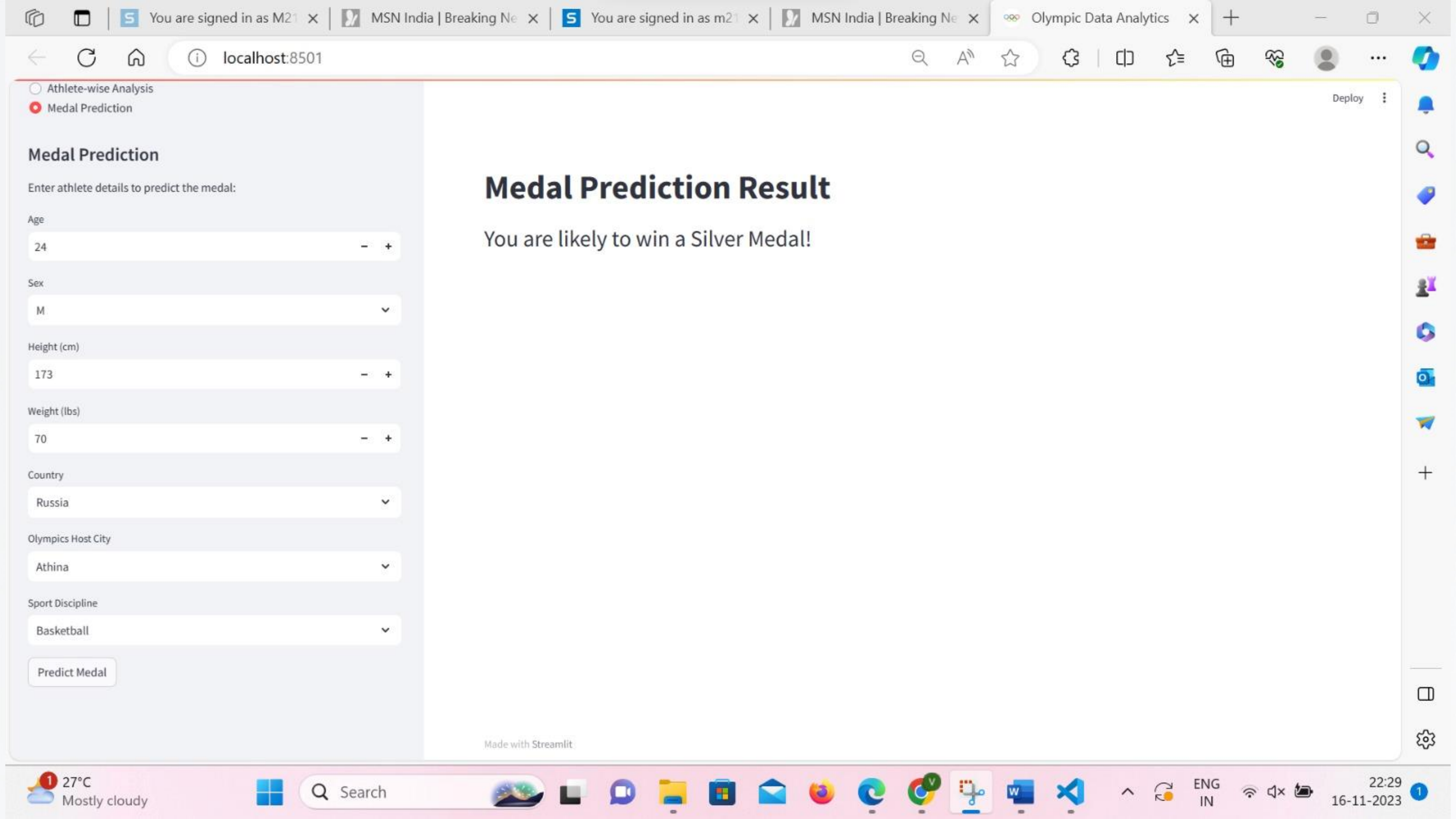
Editions	Hosts	Sports
28	23	52

Events	Nations	Athletes
651	206	116122

Participating Nations over the years


localhost:8501
Deploy
Olympics Data Analysis
Select an Option
Medal Tally
Overall Analysis
Country-wise Analysis
Athlete-wise Analysis
Medal Prediction
Country-wise Analysis
Select a Country
USA
USA excels in the following sports
Archery
Art Competitions
Athletics
Baseball
Basketball
Beach Volleyball
Boxing
Canoeing
Cycling
Diving
Equestrianism
Fencing
Figure Skating
Football
Golf
Gymnastics
Hockey
Ice Hockey
Jeu De Paume
Judo
Lacrosse
Modern Pentathlon
Polo
Roque
Rowing
Rugby
Sailing
Shooting
Sport
60
50
40
30
20


localhost:8501
Deploy
Olympics Data Analysis
Select an Option
Medal Tally
Overall Analysis
Country-wise Analysis
Athlete-wise Analysis
Medal Prediction
Distribution of Age w.r.t Sports(Gold Medalist)
Gold Medalist
Age
Ice Hockey
Polo
Rugby
Triathlon
Beach Volleyball
Rugby Sevens
Rhythmic Gymnastics
Baseball
Table Tennis
Synchronized Swimming
Volleyball
Archery
Softball
Golf
Tennis
Canoeing
Diving
Cycling
Taekwondo
Boxing
Shooting
Fencing
Height Vs Weight
Select a Sport
Overall


You are signed in as M21
MSN India | Breaking Ne
You are signed in as m21
MSN India | Breaking Ne
Olympic Data Analytics
localhost:8501
Athlete-wise Analysis
Medal Prediction
Deploy
Medal Prediction
Enter athlete details to predict the medal:
Age
24
Sex
M
Height (cm)
173
Weight (lbs)
70
Country
Russia
Olympics Host City
Athina
Sport Discipline
Basketball
Predict Medal
Medal Prediction Result
You are likely to win a Silver Medal!
Made with Streamlit
27°C
Mostly cloudy
Search
ENG
IN
22:29
16-11-2023

CONCLUSION

"Olympic Data Analysis with Prediction" is a valuable project that uses data analytics and machine learning to understand the Olympic Games and predict future medal outcomes.

It benefits sports enthusiasts, data scientists, Olympic committees, and students/researchers.

It leverages the power of data to provide valuable insights into the Olympics..

REFERENCES

1.DATASET :https://www.kaggle.com/datasets/heesoo37/120-years-of-olympic-history-athletes-and-results

2. PAPER 1 :https://ieeexplore.ieee.org/abstract/document/10010351/

3. PAPER 2 : https://dl.acm.org/doi/abs/10.1145/3404512.3404513

4. PAPER 3: https://www.sciencedirect.com/science/article/pii/S0040162521007459

5. SITES : https://www.geeksforgeeks.org/machine-learning/

THANK YOU