OLYMPIC MEDALS PREDICTION

Detailed report

ABSTRACT

This project explores the use of machine learning (ML) and deep learning (DL) techniques to predict the total number of Olympic medals a country will win.

Hriddhi Doley
Data Science

Introduction

This project explores the use of machine learning (ML) and deep learning (DL) techniques to predict the total number of Olympic medals a country will win. By analyzing various factors like GDP, population, and sports infrastructure, this study provides valuable insights into the key determinants of Olympic success. Leveraging models such as Random Forest and Neural Networks, the findings highlight the interplay between economic and sports-related features in influencing medal counts.

Understanding the data

The dataset includes the following features:

- **iso**: Country ISO code
- **ioc**: International Olympic Committee code
- **name**: Country name
- **continent**: Continent of the country
- **population**: Population of the country
- **gdp**: Gross Domestic Product (GDP)
- **sports_index**: Indicator of sports infrastructure and support
- **olympics_index**: Calculated index related to Olympic performance
- **total**: Total number of medals won

The target variable is the total number of medals won by each country.

How big is the data?

```
[3]  # Shape of the dataset
     print("Dataset Shape:", data.shape)
```

```
Dataset Shape: (93, 14)
```

How does the data look like?

```
[4] # First few rows
    print("First 5 rows:")
    print(data.head())
```

```
First 5 rows:
  iso  ioc        name      continent  population           gdp  \
0  ARG  ARG   Argentina  South America    45376763   383066977654
1  ARM  ARM     Armenia           Asia     2963234    12645459214
2  AUS  AUS   Australia        Oceania    25687041  1330900925057
3  AUT  AUT     Austria         Europe     8917205   428965397959
4  AZE  AZE  Azerbaijan         Europe    10110116    42607176471

   olympics_index  sports_index  olympicsIndex  sportsIndex  total  gold  \
0       19.597142      9.324537      19.597142     9.324537      3     0
1       19.681457     13.497324      19.681457    13.497324      4     0
2       31.170099     11.073845      31.170099    11.073845     46    17
3       12.212139     15.923033      12.212139    15.923033      7     1
4       18.213838     13.103344      18.213838    13.103344      7     0

   silver  bronze
0       1       2
1       2       2
2       7      22
3       1       5
4       3       4
```

```
[5] # Random sample of 5 rows
    print("Random Sample:")
    print(data.sample(5))
```

```
Random Sample:
    iso  ioc        name      continent  population           gdp  \
87  UKR  UKR     Ukraine         Europe    44134693  155582008717
89  UZB  UZB  Uzbekistan           Asia    34232050   57707189945
61  MYS  MAS    Malaysia           Asia    32365998  336664444247
36  GRD  GRN     Grenada  North America      112519    1089203704
4   AZE  AZE  Azerbaijan         Europe    10110116   42607176471

    olympics_index  sports_index  olympicsIndex  sportsIndex  total  gold  \
87       18.783582     14.344996      18.783582    14.344996     19     1
89       24.830288     11.631212      24.830288    11.631212      5     3
61       10.995892      8.920692      10.995892     8.920692      2     0
36       30.034643     15.971362      30.034643    15.971362      1     0
4        18.213838     13.103344      18.213838    13.103344      7     0

    silver  bronze
87       6      12
89       0       2
61       1       1
36       0       1
4        3       4
```

What is the data type of cols?

```
# Dataset Info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 93 entries, 0 to 92
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   iso             93 non-null     object
 1   ioc             93 non-null     object
 2   name            93 non-null     object
 3   continent       88 non-null     object
 4   population      93 non-null     int64
 5   gdp             93 non-null     int64
 6   olympics_index  91 non-null     float64
 7   sports_index    91 non-null     float64
 8   olympicsIndex   93 non-null     float64
 9   sportsIndex     93 non-null     float64
 10  total           93 non-null     int64
 11  gold            93 non-null     int64
 12  silver          93 non-null     int64
 13  bronze          93 non-null     int64
dtypes: float64(4), int64(6), object(4)
memory usage: 10.3+ KB
```

Are there any missing values?

```
# Null values
print("Missing Values:")
print(data.isnull().sum())
```

```
Missing Values:
iso               0
ioc               0
name              0
continent         5
population        0
gdp               0
olympics_index    2
sports_index      2
olympicsIndex     0
sportsIndex       0
total             0
gold              0
silver            0
bronze            0
dtype: int64
```

How does the data look mathematically?

```
# Statistical summary
print("Statistical Summary:")
print(data.describe())
```

```
Statistical Summary:
         population           gdp  olympics_index  sports_index  \
count  9.300000e+01  9.300000e+01       91.000000     91.000000
mean   6.639237e+07  8.668410e+11       20.677422     16.329262
std    2.057474e+08  2.702387e+12       12.631319      8.932897
min    3.393800e+04  0.000000e+00        1.000000      7.396478
25%    4.994724e+06  4.369766e+10       12.694592     10.823462
50%    1.132662e+07  1.698354e+11       18.783582     13.931504
75%    4.735157e+07  5.153325e+11       26.099431     19.019704
max    1.402112e+09  2.093660e+13      100.000000     72.227313

       olympicsIndex  sportsIndex       total       gold     silver     bronze
count      93.000000    93.000000   93.000000  93.000000  93.000000  93.000000
mean       20.232746    15.978095   11.612903   3.655914   3.634409   4.322581
std        12.852103     9.150623   19.091332   7.022471   6.626339   6.210372
min         0.000000     0.000000    1.000000   0.000000   0.000000   0.000000
25%        12.212139    10.607469    2.000000   0.000000   0.000000   1.000000
50%        18.213838    13.891772    4.000000   1.000000   1.000000   2.000000
75%        26.037386    18.984764   11.000000   3.000000   4.000000   5.000000
max       100.000000    72.227313  113.000000  39.000000  41.000000  33.000000
```

Are there any duplicate values?

```
[11] # Duplicated rows
     print("Number of Duplicated Rows:", data.duplicated().sum())
```

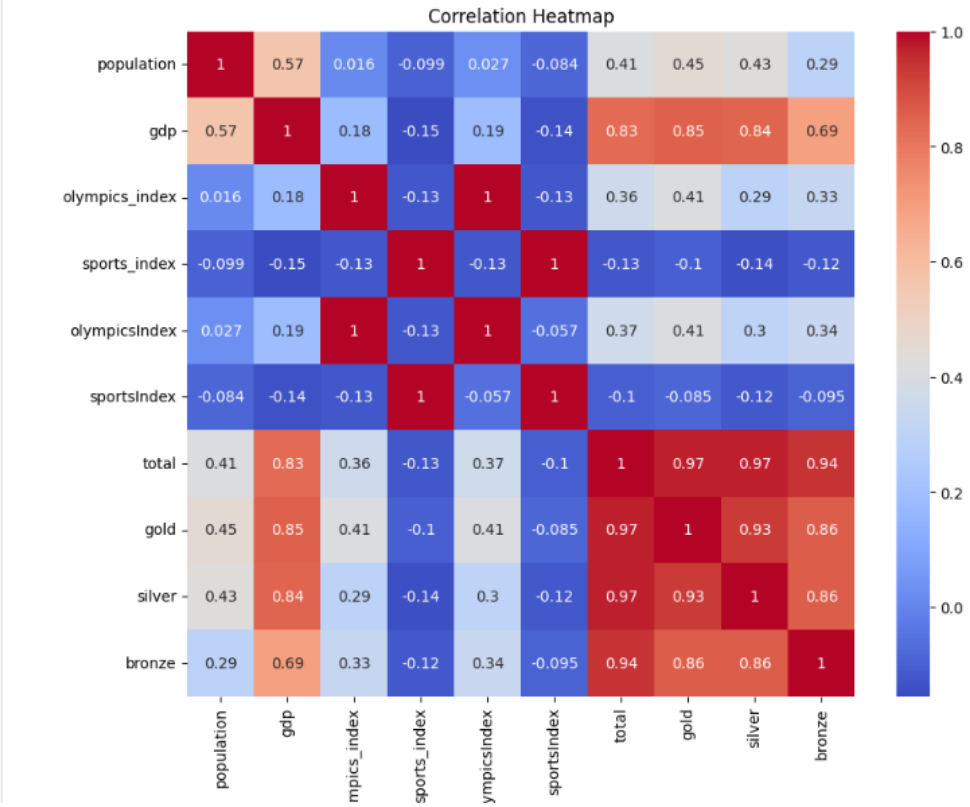
```
Number of Duplicated Rows: 0
```

```
# Drop duplicates
data.drop_duplicates(inplace=True)
```

How is the correlation between cols?

```
# Correlation matrix
print("Correlation Matrix:")
plt.figure(figsize=(10, 8))
# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=np.number)
sns.heatmap(numerical_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

```
Correlation Matrix:
```

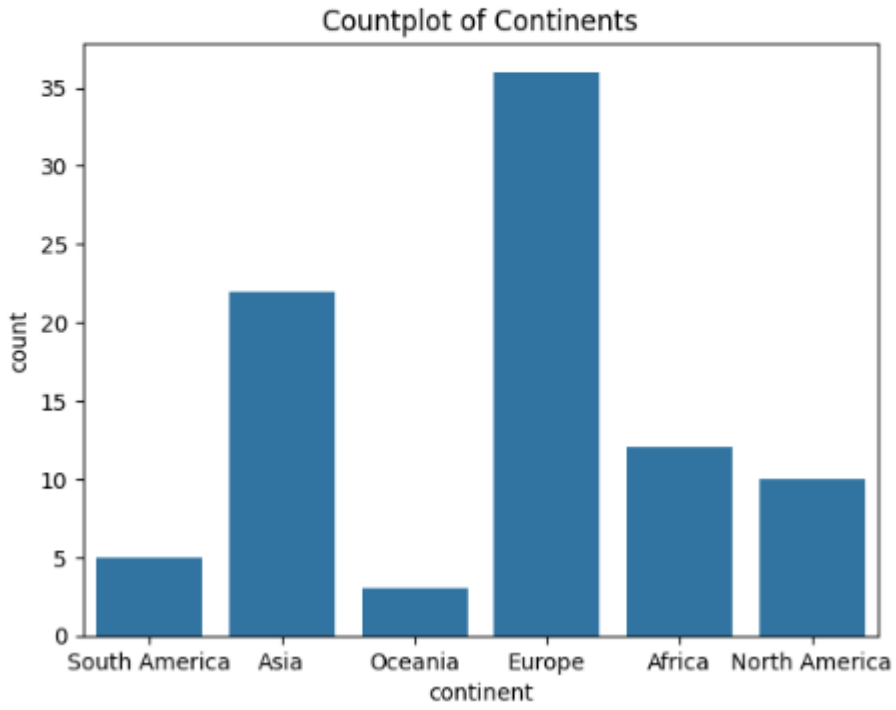


Exploratory Data Analysis

Univariate Analysis

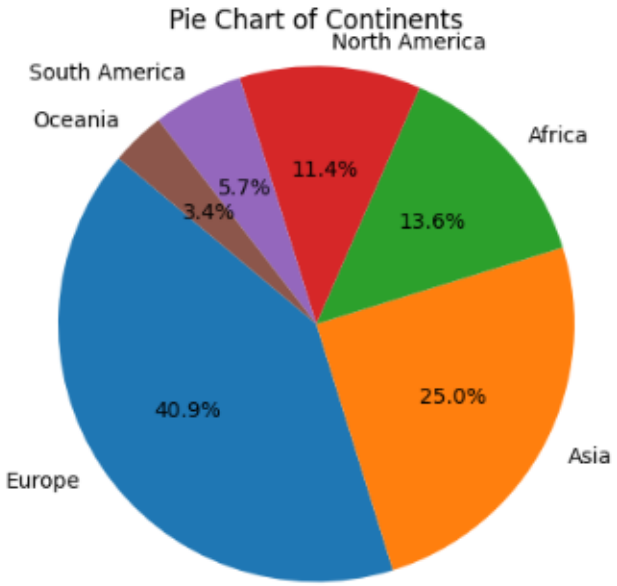
1. Countplot

```python
# Countplot for a categorical variable (continent)
sns.countplot(x='continent', data=data)
plt.title('Countplot of Continents')
plt.show()
```



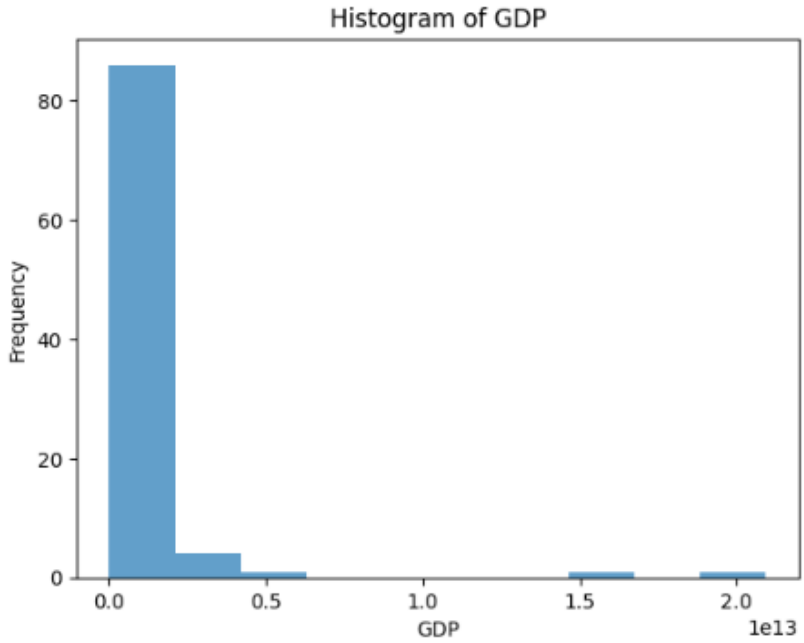
2. Piechart

```
# Pie chart for total medals distribution
labels = data['continent'].value_counts().index  # Get labels from value_counts index
sizes = data['continent'].value_counts()
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart of Continents')
plt.axis('equal')
plt.show()
```



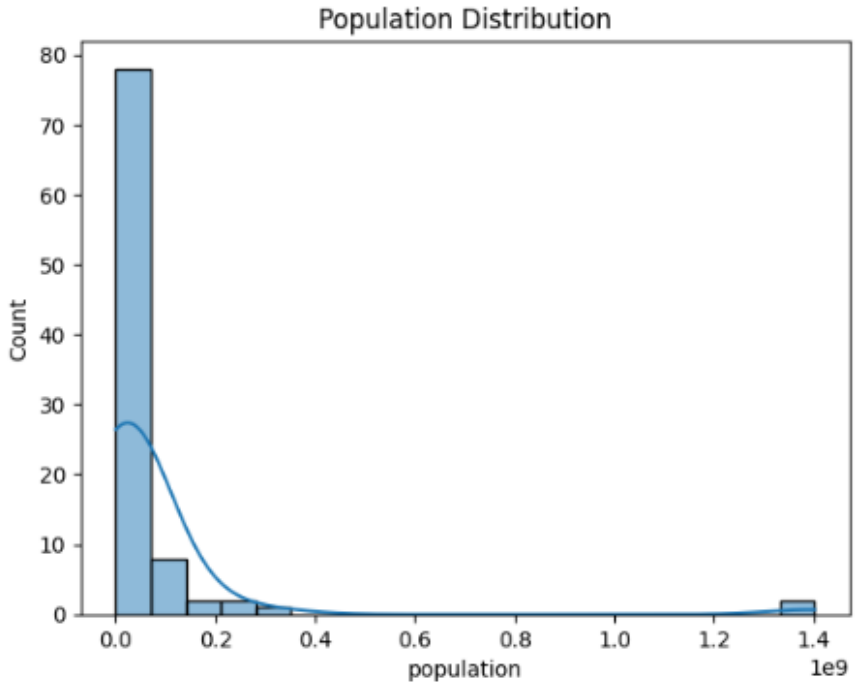
3. Histogram

```
# Histogram for GDP
plt.hist(data['gdp'], bins=10, alpha=0.7)
plt.title('Histogram of GDP')
plt.xlabel('GDP')
plt.ylabel('Frequency')
plt.show()
```



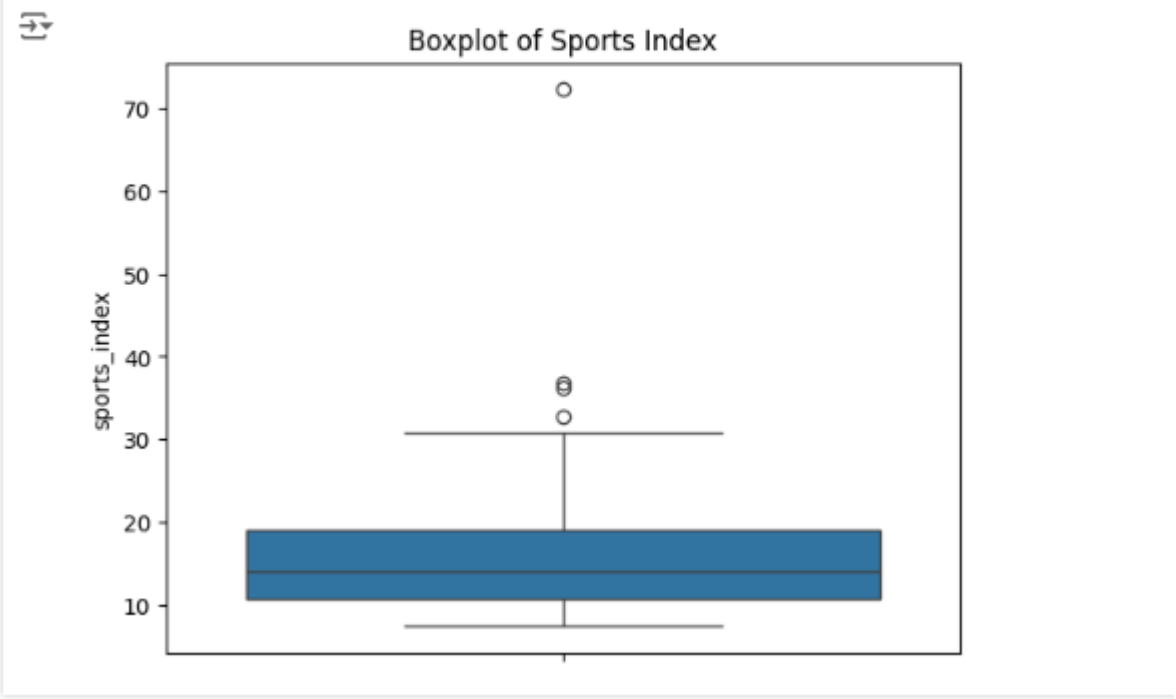
4. Distplot

```
# Distplot for population
sns.histplot(data['population'], kde=True, bins=20)
plt.title('Population Distribution')
plt.show()
```



5. Boxplot

```
# Boxplot for sports index
sns.boxplot(y='sports_index', data=data)
plt.title('Boxplot of Sports Index')
plt.show()
```



6. Min(), max(), mean(), std dev, variance

```
[20] # Summary statistics for specific features
     print("GDP Statistics:")
     print("Min:", data['gdp'].min())
     print("Max:", data['gdp'].max())
     print("Mean:", data['gdp'].mean())
     print("Standard Deviation:", data['gdp'].std())
     print("Variance:", data['gdp'].var())
```

```
GDP Statistics:
Min: 0
Max: 20936600000000
Mean: 866840997410.7097
Standard Deviation: 2702387141886.764
Variance: 7.302896264634914e+24
```

7. Skewness

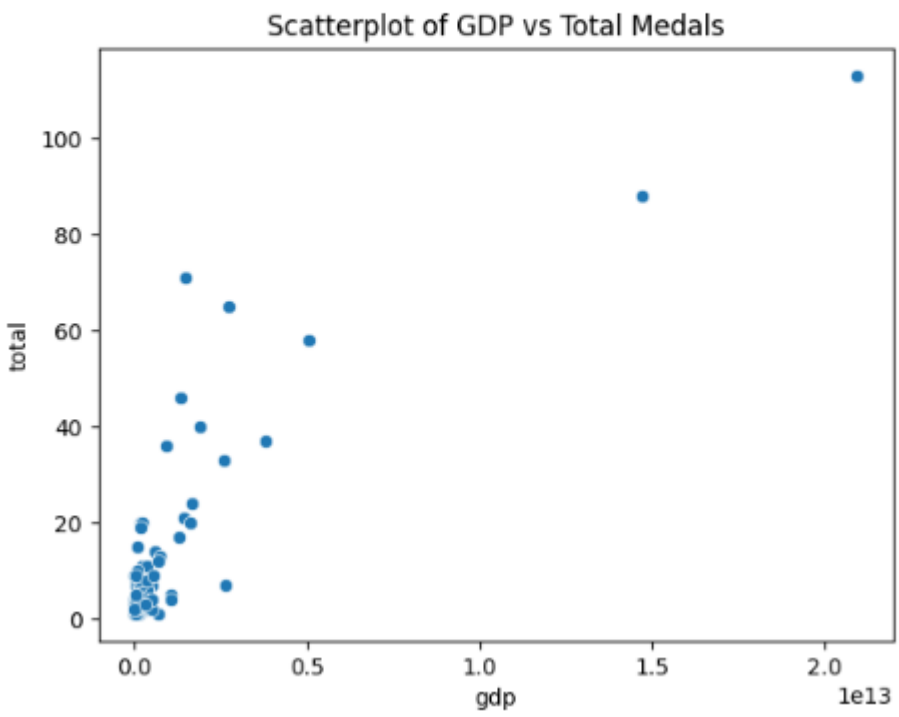
```python
# Skewness of numerical columns
print("Skewness of Numerical Features:")
print(data[['gdp', 'population', 'sports_index']].skew())
```

```
Skewness of Numerical Features:
gdp             6.098092
population      5.958251
sports_index    3.188289
dtype: float64
```

Multivariate Analysis

Scatterplot

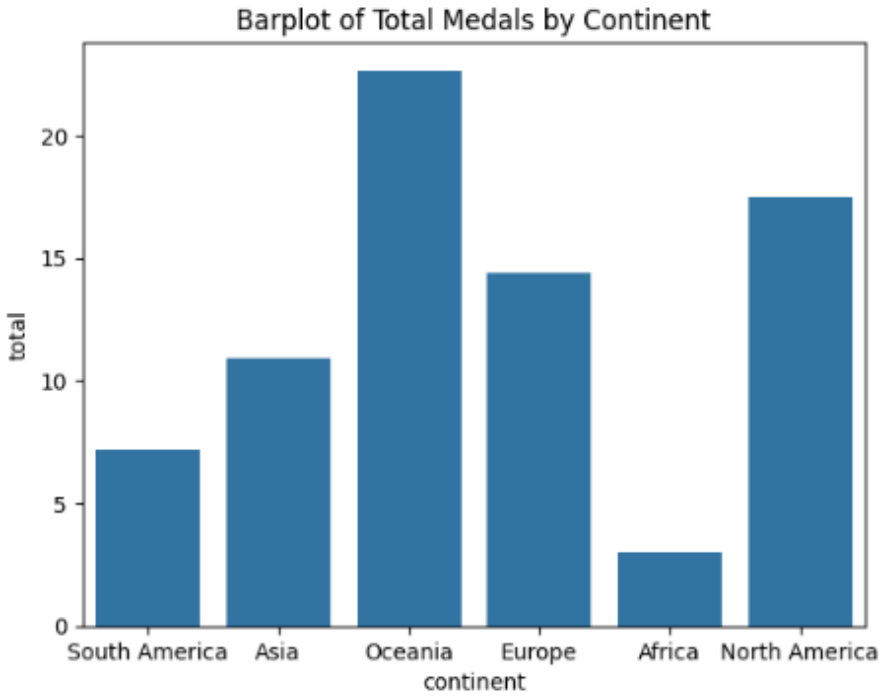
```python
# Scatterplot for GDP vs. Total Medals
sns.scatterplot(x='gdp', y='total', data=data)
plt.title('Scatterplot of GDP vs Total Medals')
plt.show()
```



Barplot

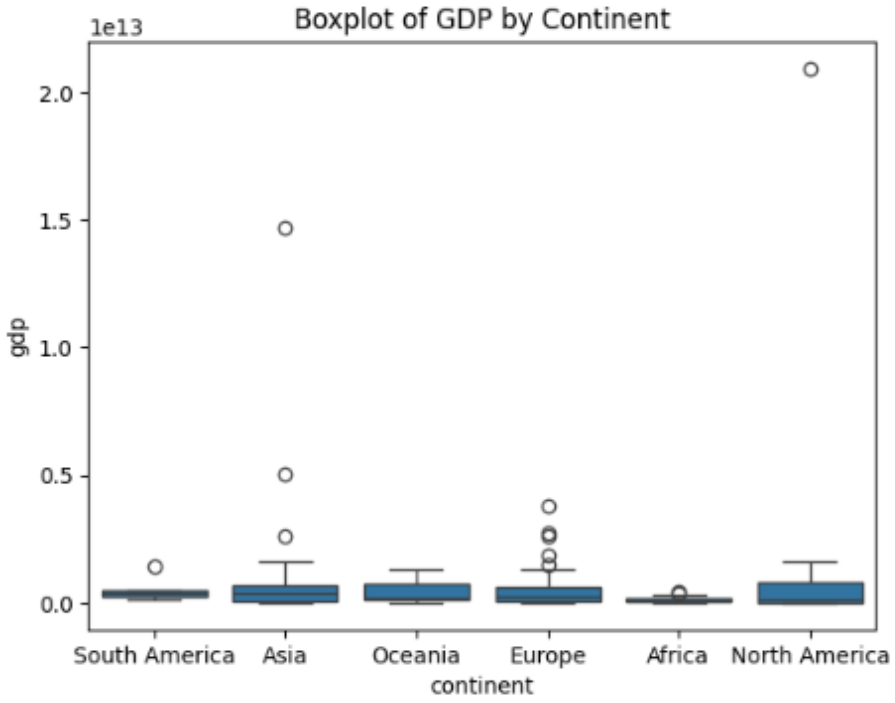
```
[23] # Barplot for medals by continent
     sns.barplot(x='continent', y='total', data=data, ci=None)
     plt.title('Barplot of Total Medals by Continent')
     plt.show()
```

```
<ipython-input-23-49069b5224a7>:2: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

  sns.barplot(x='continent', y='total', data=data, ci=None)
```



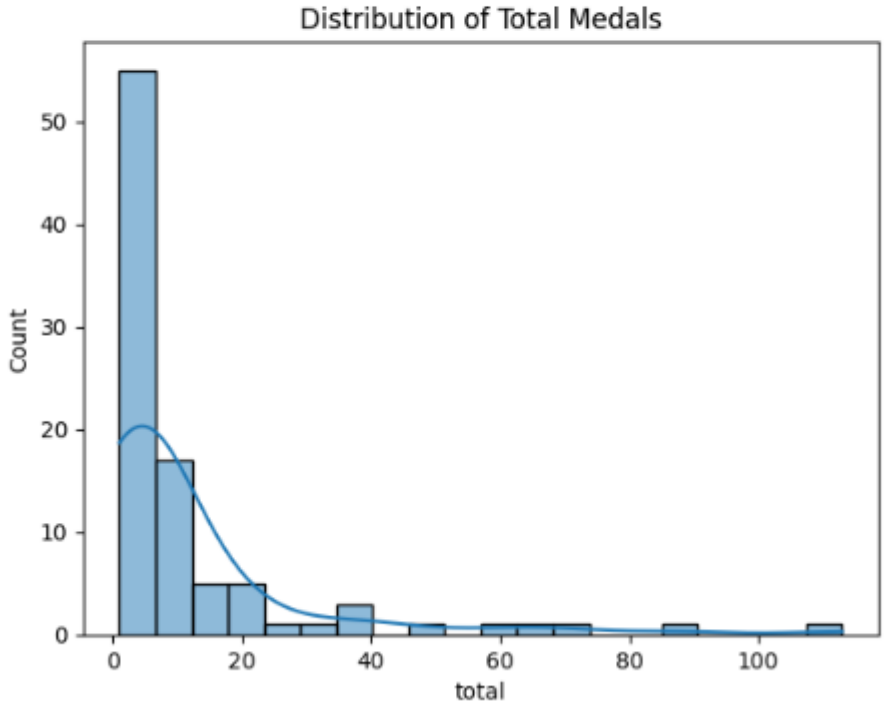
Boxplot

```
# Boxplot for GDP by continent
sns.boxplot(x='continent', y='gdp', data=data)
plt.title('Boxplot of GDP by Continent')
plt.show()
```



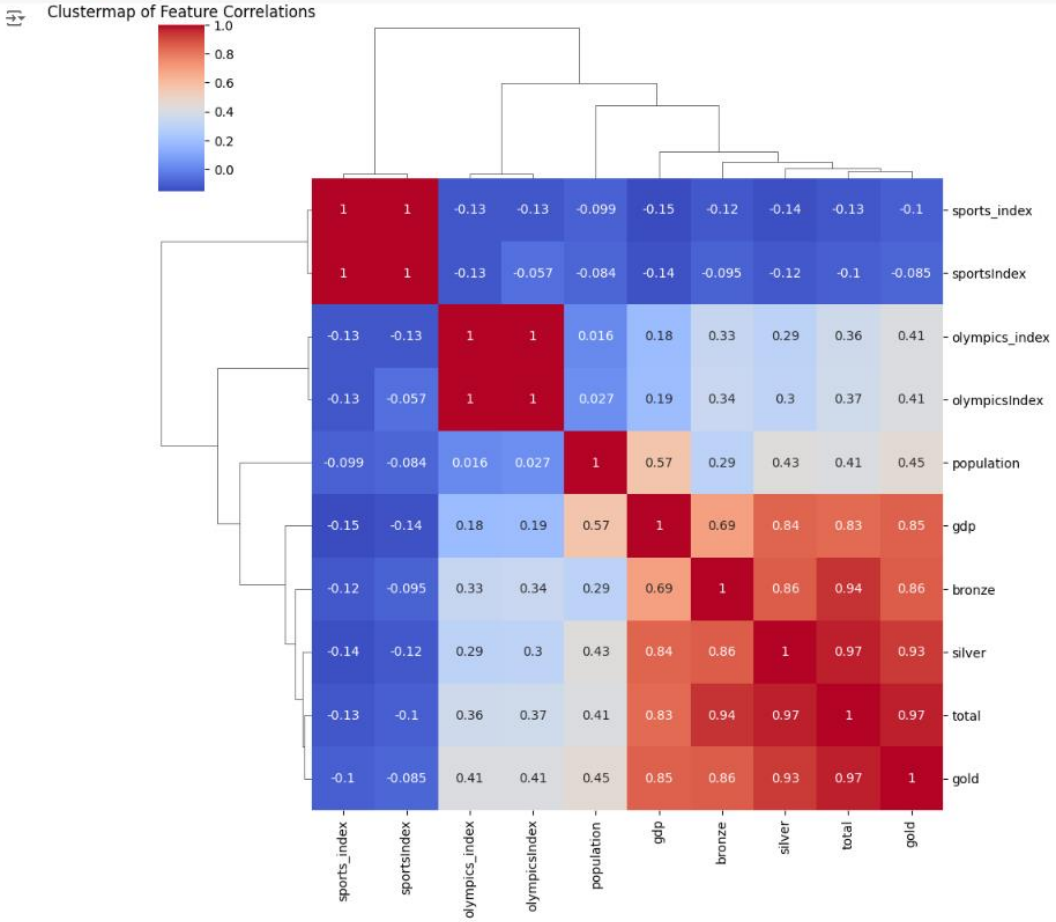
Distplot

```
[26] # Distplot for Total Medals
     sns.histplot(data['total'], kde=True, bins=20)
     plt.title('Distribution of Total Medals')
     plt.show()
```



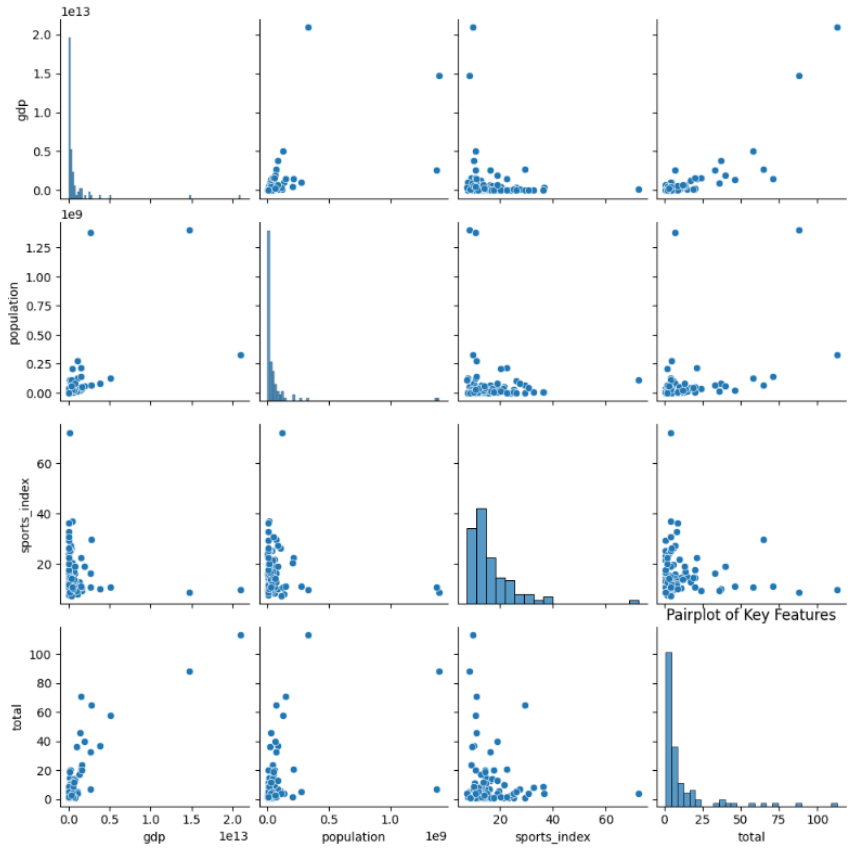
Clustermap

```
# Clustermap of correlations
# Select only numerical features for correlation calculation
numerical_data = data.select_dtypes(include=np.number)
sns.clustermap(numerical_data.corr(), cmap='coolwarm', annot=True)
plt.title('Clustermap of Feature Correlations')
plt.show()
```



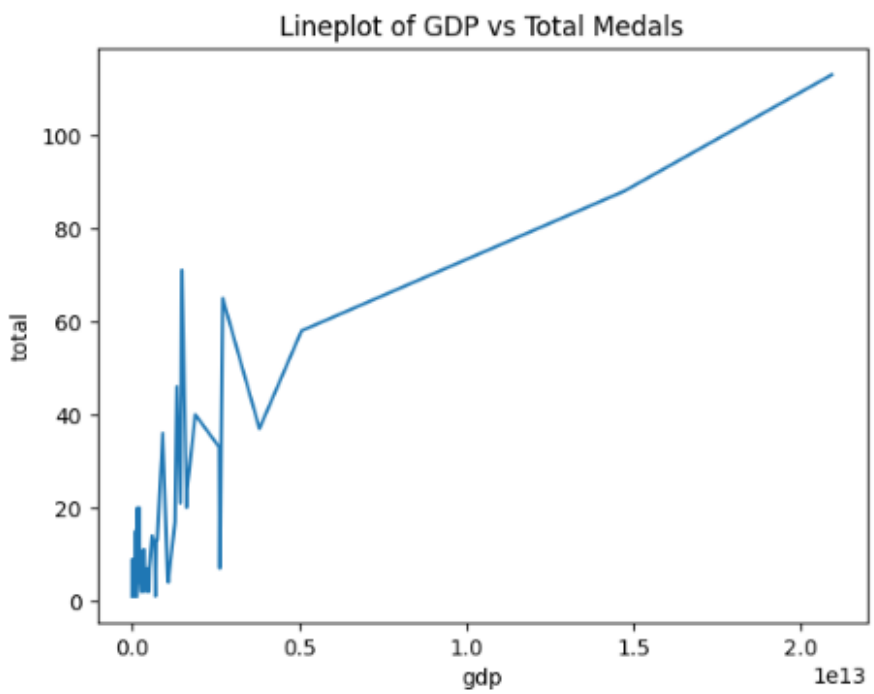
Pairplot

```
# Pairplot for key numerical features
sns.pairplot(data[['gdp', 'population', 'sports_index', 'total']])
plt.title('Pairplot of Key Features')
plt.show()
```



lineplot

```
# Lineplot for GDP and Total Medals
sns.lineplot(x='gdp', y='total', data=data)
plt.title('Lineplot of GDP vs Total Medals')
plt.show()
```



Data Preprocessing

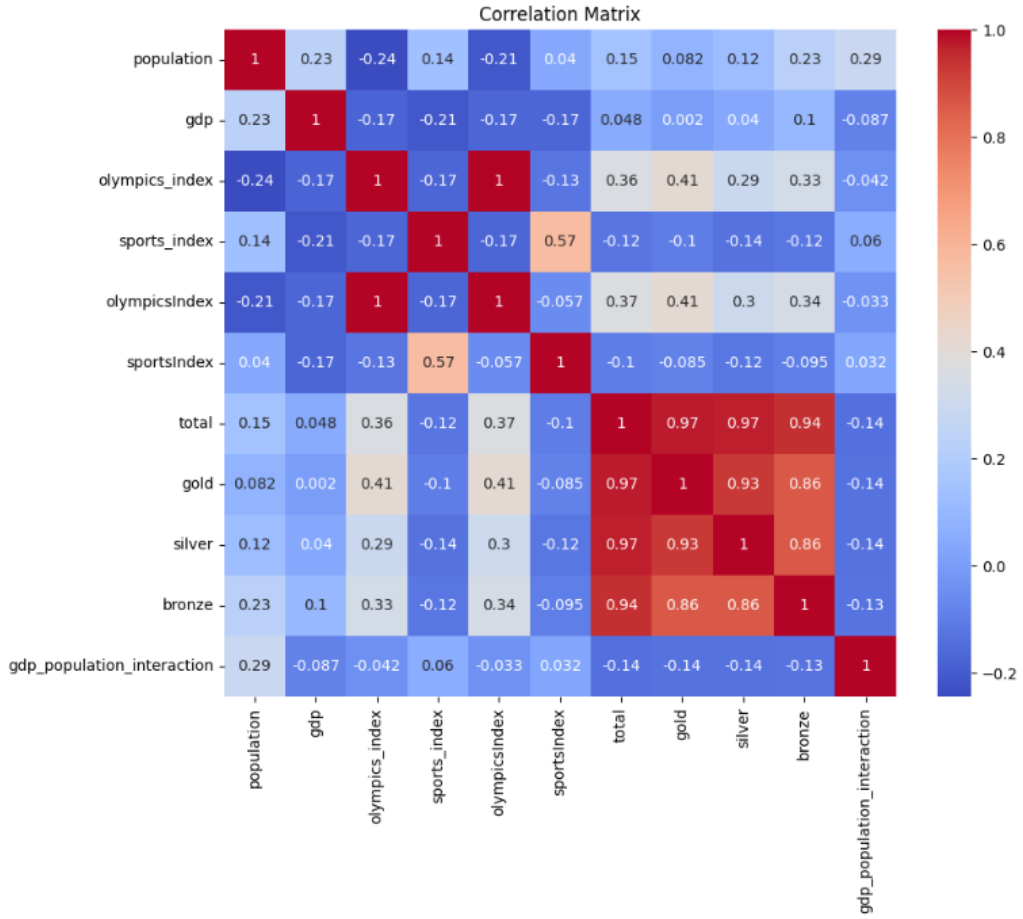
1. **Missing Value Imputation**: Missing values in numerical features were imputed using the mean.
2. **Categorical Features:** The 'continent' column was converted to numerical using one-hot encoding.
3. **Outlier Handling**: Outliers were identified using the IQR method and replaced with the median value.
4. **Feature Scaling**: Features were standardized using StandardScaler.
5. **Feature Construction**: An interaction term 'gdp_population_interaction' was created
6. **Advanced Feature Selection**:
   - **SelectKBest:** Top 3 features were selected using ANOVA F-statistics.
   - **PCA**: Reduced feature space to two principal components.

EDA post data processing

1. **Descriptive Statistics**:
    - TBC
2. **Visualizations**:

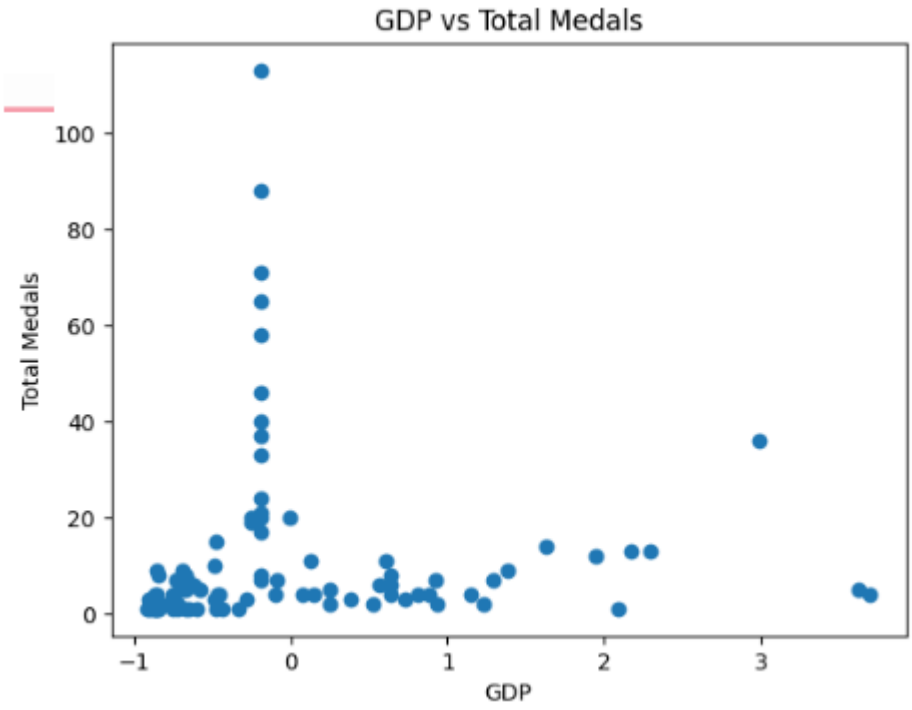
Correlation Heatmap:

1. **Sports Index:** A critical feature for predicting total medals due to its significant correlation with total medals and its association with other performance indicators like olympics_index.
2. **Olympics Index:** Strongly correlated with total medals and gold medals, making it another valuable feature for predictive models.
3. **Gold, Silver, and Bronze Medals:**As components of total medals, these are inherently strong predictors but cannot be used in isolation for predictive modeling unless broken down by feature or year.
4. **GDP and Population:**While these features have weak direct correlations with total medals, interaction terms (e.g., gdp_population_interaction) may capture their combined influence effectively.



Scatter Plot:

1. **GDP Alone is Insufficient:**
    a. While GDP may indirectly influence performance (e.g., through its correlation with sports infrastructure), it is not a standalone predictor of total medals.
    b. Interaction terms like gdp_population_interaction or additional features like sports index are essential to capture this relationship.
2. **Focus on Outliers:**
    a. Investigate the countries with high medals but varying GDP levels to identify other common factors (e.g., training programs, government policies, historical performance).



Machine Learning Models

Three machine learning models were implemented to predict the total number of medals:

1. **Linear Regression**:
    - A simple model that assumes a linear relationship between the input features and the target variable.
    - Used as a baseline for comparison.
2. **Decision Tree**:
    - Captures non-linear relationships by splitting the data into subsets based on feature values.
    - Handles interactions between features effectively but can overfit if not regularized.
3. **Random Forest**:
    - An ensemble model combining multiple decision trees.
    - Improves robustness and reduces overfitting by averaging the predictions from individual trees.
    - Tends to perform well with complex, non-linear relationships.

Implementation Steps

1. **Training the Models**:
    - Models were trained on the **training dataset** (X_train and y_train).
    - Each model used default or basic configurations to provide initial predictions (y_pred).
2. **Evaluation Function**:
    - A custom evaluation function was defined to calculate three metrics:
        - **Mean Absolute Error (MAE)**: Measures the average absolute differences between predicted and actual values.
        - **Mean Squared Error (MSE)**: Penalizes larger errors more heavily than MAE.
        - **R-squared (R^2)**: Measures the proportion of variance in the target explained by the model.
3. **Model Evaluation**:
    - Each model's predictions (y_pred_lr, y_pred_dt, y_pred_rf) were compared against the **testing dataset** (y_test).

- Evaluation metrics were computed and stored in a results dataframe for comparison.

Advantages of this Methodology

1. **Model Comparisons**:
    - Using multiple models provides a benchmark and identifies the most suitable approach for the dataset.
2. **Error Metrics**:
    - Comprehensive evaluation with MAE, MSE, and R^2 offers a clear understanding of model performance.
3. **Ensemble Model**:
    - Random Forest improves prediction reliability by averaging multiple trees, mitigating overfitting.

Future Improvement

- **Hyperparameter Tuning**:
    - Optimize model parameters (e.g., max_depth, n_estimators) for Decision Tree and Random Forest.
- **Feature Importance**:
    - Use Random Forest to identify the most influential features.
- **Cross-Validation**:
    - Incorporate K-Fold Cross-Validation for robust performance evaluation.

Deep Learning Model

1. **Neural Network Architecture**:
    - **Input Layer**: Accepts numerical features with dimensions matching X_train.
    - **Hidden Layers**: Two layers with 32 and 64 neurons, activated by ReLU.
    - **Output Layer**: Single neuron for regression output.
2. **Training and Validation**:
    - **Optimizer**: Adam.
    - **Loss Function**: Mean Squared Error (MSE).
    - **Epochs**: 50 with a batch size of 32.
    - **Validation Split**: 20% of training data used for validation.
3. **Performance**:
    - **Loss (MSE)**: 57.1789 on the test dataset.
    - **Mean Absolute Error (MAE)**: 5.6227, indicating an average deviation of 5.62 medals.
    - **Training History**: Consistent reduction in training and validation loss over 50 epochs.
    - Demonstrated improved accuracy compared to traditional ML models.

Potential Improvements

1. **Regularization**:
    - Add Dropout layers to prevent overfitting by randomly disabling neurons during training.
2. **Hyperparameter Tuning**:
    - Experiment with:
        - Learning rates.
        - Number of layers and neurons.
        - Activation functions (e.g., Leaky ReLU).
3. **Early Stopping**:
    - Monitor validation loss and stop training when it stops improving.
4. **Cross-Validation**:
    - Use K-Fold Cross-Validation to ensure robust evaluation across multiple splits.

Model Evaluation and Results

Model	MAE	MSE	R^2
Linear Regression	8.982610	104.580419	-2.075895
Decision Tree	8.210526	127.894737	-2.761610
Random Forest	8.061053	106.149516	-2.122045
Neural Network	5.622690	57.178936	-0.681733

Findings and Insights

1. **Neural Network Performance**:
    - The **Neural Network** outperformed all traditional models, achieving the lowest MAE (5.62) and MSE (57.17).
    - It demonstrated its ability to capture complex, non-linear relationships in the dataset.
2. **Traditional Models**:
    - **Linear Regression**:
        - Performed the worst, with the highest MAE (8.98) and MSE (104.58).
        - The inability to model non-linear relationships led to subpar performance.
    - **Decision Tree**:
        - Showed slightly better results than Linear Regression but suffered from overfitting, reflected in high MSE (127.89).
    - **Random Forest**:
        - Performed better than both Linear Regression and Decision Tree with MAE (8.06), showcasing its strength in handling feature interactions.
3. **General Observations**:
    - R^2 values for all models were negative, indicating poor goodness-of-fit. This suggests the presence of non-linear patterns in the data that traditional models struggled to capture effectively.

Feature Importance

Top Features:

- **Sports Index (sports_index)** and **Olympics Index (olympics_index)**:
    - Strongly correlated with total medals, making them critical predictors.
- **GDP and Population**:

    - Weak individual correlations with total medals, but their interaction term (gdp_population_interaction) may provide additional predictive value.

- **Neural Network**:

  Able to leverage the combined influence of all features, including non-linear interactions, to achieve superior performance.

Conclusion

1. **Key Takeaway**:
    - Neural Networks are highly effective in capturing non-linear relationships and interactions, making them the most suitable model for predicting total Olympic medals.
    - Traditional models like Linear Regression and Decision Trees struggled due to their inability to model complex patterns.
2. **Importance of Features**:
    - Sports-related indices (sports_index and olympics_index) are crucial for predicting medal counts.
    - Economic indicators (e.g., GDP, population) contribute indirectly, primarily through interaction terms.

Future Work

1. **Feature Engineering**:
    - Explore additional interaction terms and non-linear transformations to improve model accuracy.
    - Incorporate other potential features, such as historical performance data or sports funding.
2. **Hyperparameter Tuning**:
    - Optimize the Neural Network architecture (e.g., number of layers, neurons, activation functions).
    - Fine-tune parameters for Decision Tree and Random Forest to mitigate overfitting.
3. **Cross-Validation**:
    - Use K-Fold Cross-Validation to ensure robust evaluation and minimize bias in model performance.
4. **Advanced Models**:
    - Experiment with more sophisticated models like Gradient Boosting Machines (e.g., XGBoost, CatBoost).

- Test convolutional networks for data involving spatial or temporal patterns.

5. **Scalability**:
    - Extend the analysis to include data from more countries and additional Olympic years to validate model performance across diverse contexts.

Future Work

1. Experiment with additional features like historical performance and government sports funding.
2. Incorporate advanced deep learning techniques, such as recurrent neural networks, to analyze time-series data.
3. Expand the dataset to include more countries and years for improved generalization.