INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE

Documentation On

"Olympic Data Analysis"

PG-e-DBDA May 2021

Submitted By:-

1) 1308_Dipali Bamdale
2) 1310_Diksha Kolikal

Centre Coordinator:-
Mr.Prashant Karhale

Project Guide:-
Mr.Akshay Tilekar

Contents

List of figures and tables

1. Introduction

1.1 PROBLEM STATEMENT

"Olympic Data Analysis"

1.2 Abstract

The Olympics is an international sporting event. The Olympic Games have been expanding every year which can be seen by the records of the nations participating. This international sporting event where thousands of athletes from various countries compete in various sports every four years, has experienced enough growth in which we can begin to ask questions on the evolution of the Olympics based on gender participation or their performance and results based on basic biological information Participation in the event since the last Olympics Given the historical data throughout the Olympics, the odds of winning a medal could perhaps be given based on a few biological attributes of the athletes. Therefore, we decided to do exploratory data analysis so we may visualize patterns within the dataset. Furthermore, we to predict if an athlete would win a medal based on those few attributes given.

1.3 Product Scope

- Selecting the algorithm meeting requirement.
- Choosing the optimum algorithm from set of algorithm.
- Testing it vs applicants.
- Analysis of result and making changes in algorithm accordingly.

1.4 Aims & Objectives

The primary objective of this project is to analyze the Olympic dataset using python to compare the overall performance of countries and evaluate each country's contribution to the Olympics. Therefore, we decided to do exploratory data analysis so we may visualize patterns within the dataset. Furthermore, we predicted if age, height, weight, year, and season would affect on medal based on these few attributes.

2. Overall Description

2.1. Software Life Cycle Model

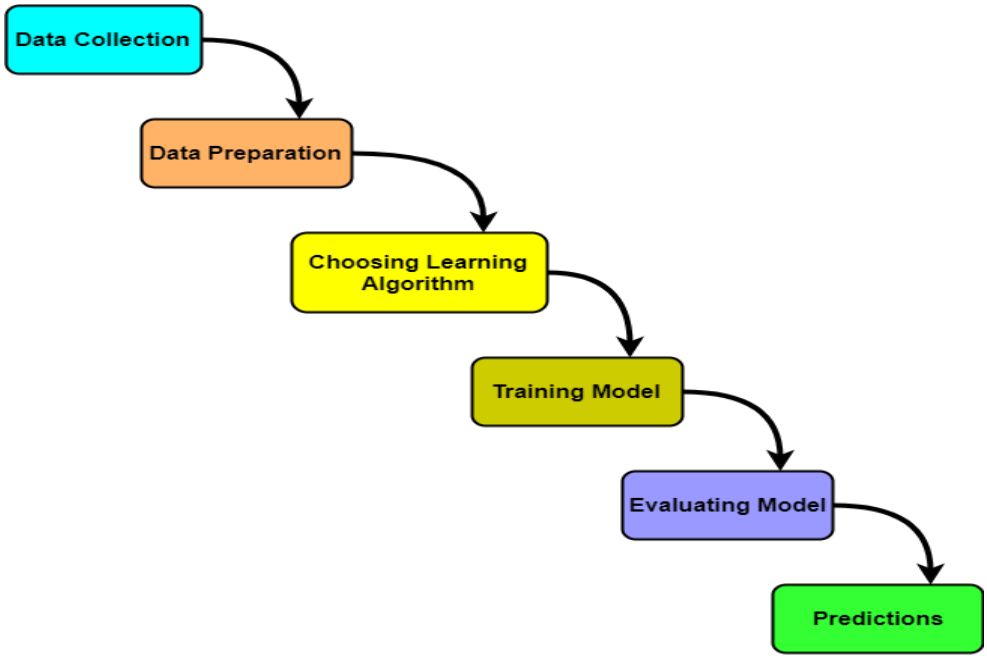


Fig.1 :-Software Life Cycle Model

1) Data Collection :

The very first step of any type of Analysis, whether it is technical or non-technical, is Data Collection. In order to perform analysis on a certain problem, we require a large amount of Data on which we apply various techniques and algorithms to reach to a particular conclusion and get our desired result. We have taken two datasets which provide us with large volume and a large variety of data for Analysis. 1st dataset consists the information about the players and their entire details like their Gender, Height, Weight, Country for which they play, Medals won (Gold, Silver and Bronze) and many more. This data can be used to analyze the performance of the particular player and can also help in the comparative study between two or more players.2nd dataset consists of the list of countries along with their country code which is the identification of these countries.

2) Data Pre-Processing:

We did further pre-processing of the data by selecting attributes we deemed relevant such as:

Sex, Age, Weight, Height, Sport, and Medal. We made the decision to remove ID, Name, Games, City and Event. These were removed based on the idea that personal identifying information would not be useful in any predictions or data analysis. The Event column was removed because it splits the Sports column based on specific games based on the sport. For example, the swimming tag would be represented as Swimming Men's 200 Meter Breaststroke, Swimming Men's 400 Meter Breaststroke, and so forth in the Event column. Therefore, we made the decision to drop the column. The dataset came with null values that had to be resolved. We identified them by checking existing null values for each column within the dataset. Our results were 9474 for Age, 60171 for Height, 62875 for Weight and 231333 for Medal. The reason the Medal column returned so many null values was because the dataset had the tags Gold, Silver, and Bronze for medallists and a null tag for non-medallists. The decision was made to give non-medallists the "No Medal" string value to make further data analysis easier. At the end of our data pre-processing step, we came out with a total of 206,165 unique athletes with the attributes: Sex, Age, Weight, Height, NOC, Year, Season, Sport, and Medal.

3) Choosing Learning Algorithm:

Our target cloumn has two types of values like medal and non-medal.So we converted that into binary form like 0 for non-medal and 1 for medal. we get the target column in binary format so we decided to apply binary classifier algorithms.And we selected Random Forest Classifier,Logistic Regression,Scalar Vector Machine and K-nearest Neighbour algorithms.

4) Training Model:

we splitted data into training and testing dataset.We build the training model.It consists of the sample output data and the corresponding sets of input data that have an influence on the output.It has learn through that dataset and has given the accuracy.

5) Evaluating Model:

Evaluating model to make it perform better way.Model Evaluation is the subsidiary part of the model development process. It is the phase that is decided whether the model performs better. Therefore, it is critical to consider the model outcomes according to every possible evaluation method. Applying different methods can provide different perspectives.We provided different parameters to our algorithms and checked it's performance according to that.And we made changes to get better performing model.

6) Predictions:

For our predictions, we wanted to compare the results of our predictions

from the various algorithms we mentioned. We plugged the data into them after a train/test split. The scores good for Random Forest Classifier, which managed to get an average score of 90%.The confusion matrix did turn out better and showed the ability to predict being different per medal, or lack there of.

2.2 Flowchart of the System

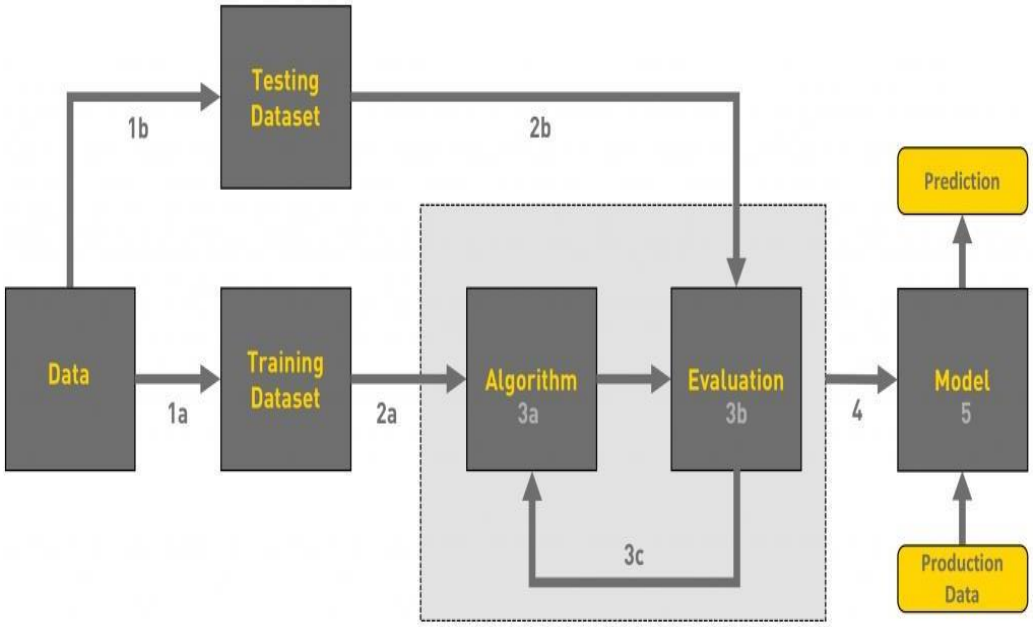


Fig.2 :-Flowchart of the system

The very first step of any type of Analysis, whether it is technical or non technical, is Data Collection. In order to perform analysis on a certain problem, we require a large amount of Data on which we apply various techniques and algorithms to reach to a particular conclusion and get our desired result.

We can't apply various techniques or Machine Learning Algorithms like Logistic Regression, Random Forest, SVM etc directly to the Raw Data. This Data need to be processed and converted into useful data. So we perform Data Pre-Processing and Exploratory Data Analysis. Then we split our dataset into training and testing dataset.

We applied Random Forest, Logistic regression, KNN and SVM algorithms on training dataset. Furthermore we did evaluation of different model accuracies and according to that we fine-tune our best model. We trained model by giving

different test datasets to our model through k-fold cross validation technique. At last we predicted the medal

2.3 Exploratory Data Analysis:

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Following are some plots we used to extract some useful information

1. we use the box plot to find out the outlier in our data forthere more we removed oulier by using iqr method.

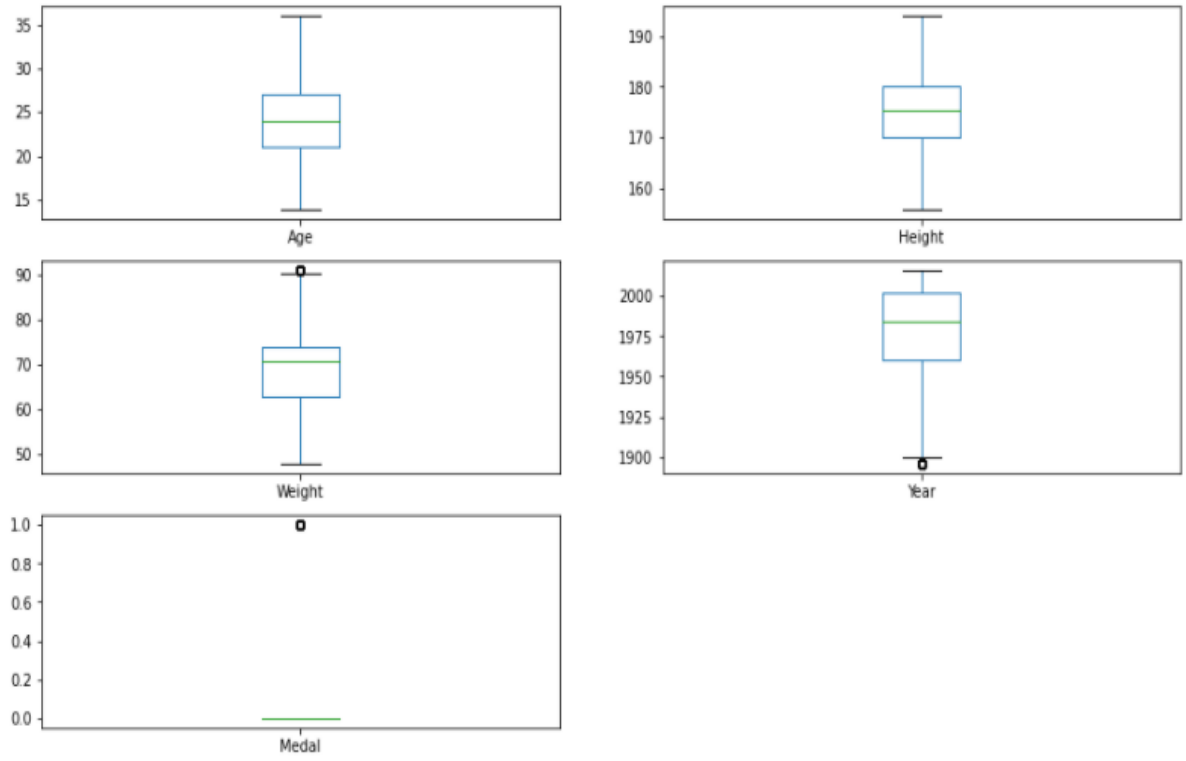


Fig.2.3.1 :- Outlier Detection

2. Graph plotted on the medal Colum by the reference of medal count and display the winning medal count and not winning medal count

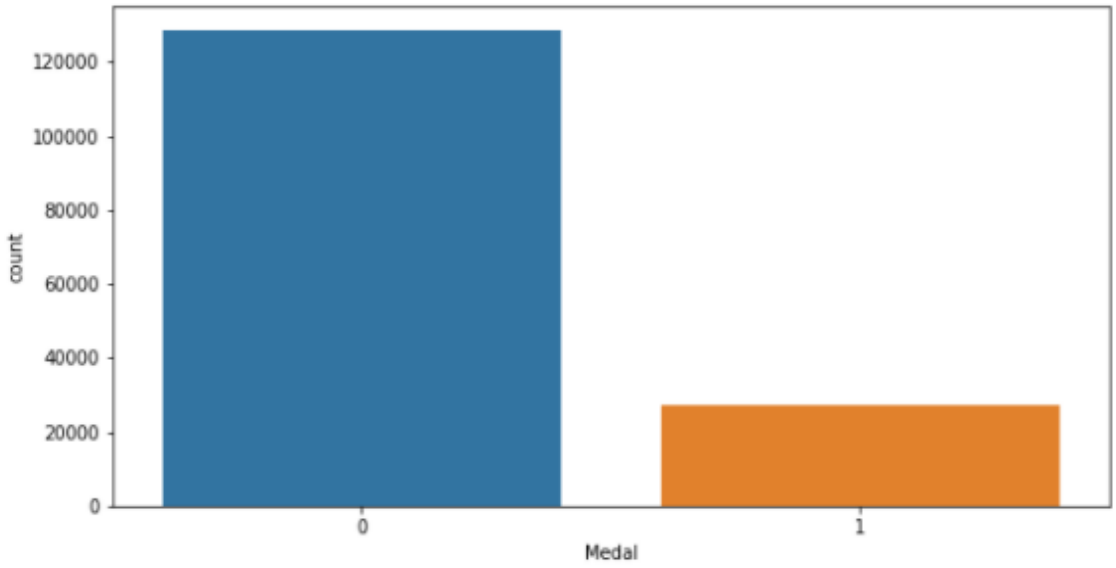


Fig.2.3.2 :- Medal Bar Chart

3.Using Gender Colum plotted the graph for displaying the participation count of male and female.

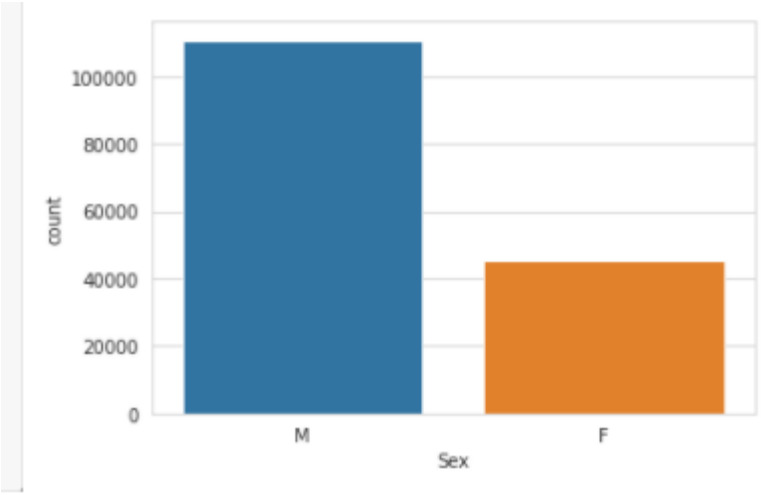


Fig.2.3.3 :- Gender Bar Chart

4. Plot graph for the showing in which season more games are arranged and we get the higher result rate of the games taken in winter season

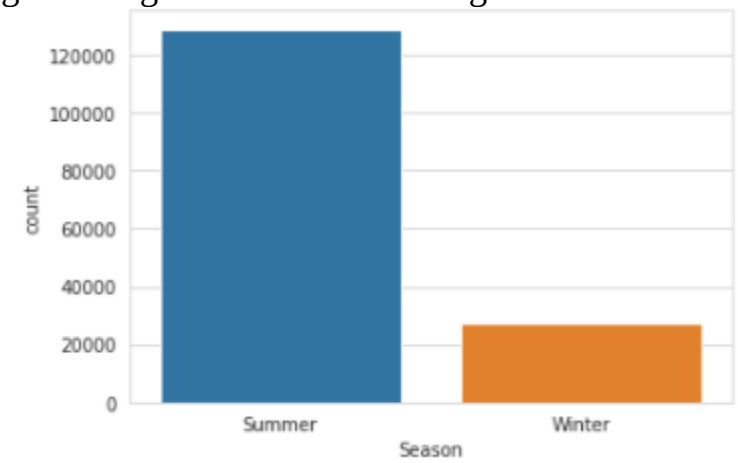


Fig.2.3.4 :- Season Bar Chart

5. Here we plot the graph for displaying top ten country's participation in the Olympic game

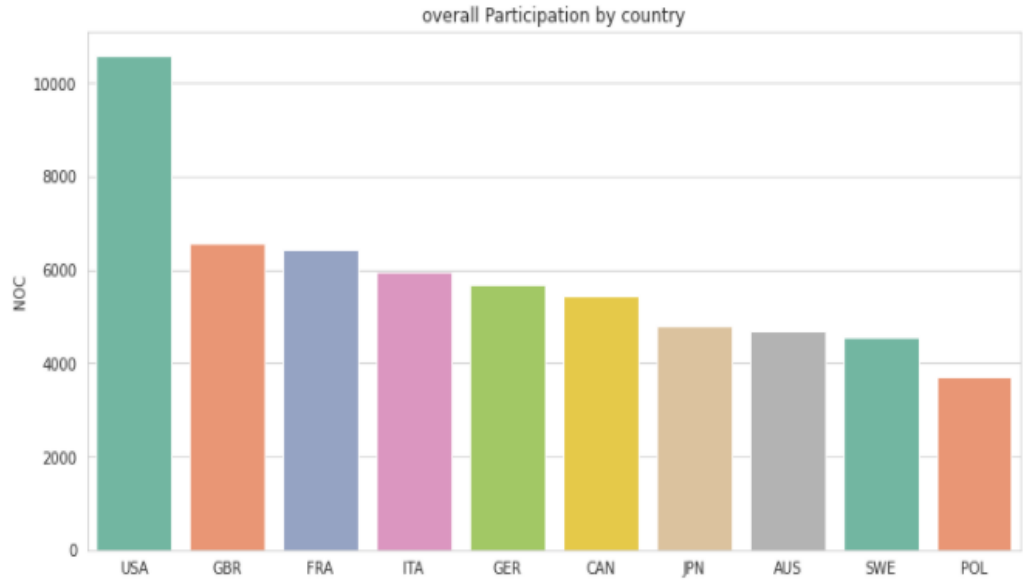


Fig.2.3.5 :- Participation Country Bar Chart

6.we use the correlation matrix to find out the ration ship between variable

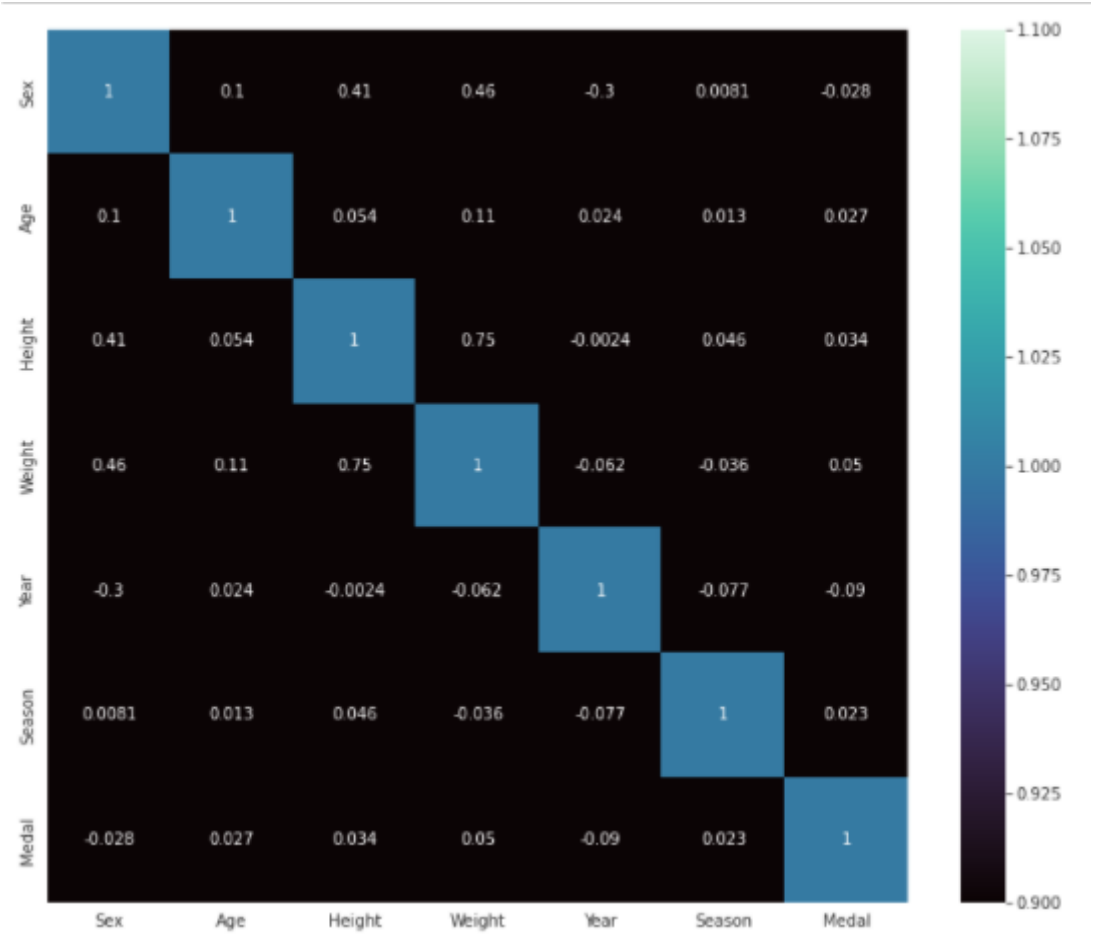


Fig.2.3.6 :- Correlation Matrix

3. **Model Building:**

1. Train/Test split:

One important aspect of all machine learning models is to determine their accuracy. Now, in order to determine their accuracy, one can train the model using the given dataset and then predict the response values for the same dataset using that model and hence, find the accuracy of the model. A better option is to split our data into two parts: first one for training our machine learning model, and second one for testing our model.

• Split the dataset into two pieces: a training set and a testing set.

• Train the model on the training set.

• Test the model on the testing set, and evaluate how well our model did.

Advantages of train/test split:

• Model can be trained and tested on different data than the one used for training.

• Response values are known for the test dataset, hence predictions can be evaluated

• Testing accuracy is a better estimate than training accuracy of out-of-sample performance.

Machine learning consists of algorithms that can automate analytical model building. Using algorithms that iteratively learn from data, machine learning models facilitate computers to find hidden insights from Big Data without being explicitly programmed where to look.
We have used the following three algorithms to build predictive model.

2. Logistic Regression:

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

```
]: 1 from sklearn.linear_model import LogisticRegression
   2 classifier= LogisticRegression(random_state=0)
   3 classifier.fit(X_train, Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWar
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

```
]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                      intercept_scaling=1, l1_ratio=None, max_iter=100,
                      multi_class='auto', n_jobs=None, penalty='l2',
                      random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                      warm_start=False)
```

```
]: 1 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
   2                   intercept_scaling=1, l1_ratio=None, max_iter=100,
   3                   multi_class='warn', n_jobs=None, penalty='l2',
   4                   random_state=0, solver='warn', tol=0.0001, verbose=0,
   5                   warm_start=False)
```

```
]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                      intercept_scaling=1, l1_ratio=None, max_iter=100,
                      multi_class='warn', n_jobs=None, penalty='l2',
                      random_state=0, solver='warn', tol=0.0001, verbose=0,
                      warm_start=False)
```

```
]: 1 #Predicting the test set result
   2 y_pred= classifier.predict(X_test)
```

```
1 #Creating the Confusion matrix
2 from sklearn.metrics import confusion_matrix,accuracy_score
3 cm= confusion_matrix(Y_test, Y_pred)
```

```
1 print("Confusion Matrix \n",cm)
```

```
Confusion Matrix
 [[22870 15587]
 [ 3532 35168]]
```

```
1 acc = accuracy_score(Y_test,Y_pred)
```

```
1 print("Accuracy : ",acc)
```

```
Accuracy :  0.7522065399121273
```

3. K nearest neighbour:

K Nearest Neighbor Algorithm. K nearest neighbor algorithm is very simple. It works based on minimum distance from the query instance to the training samples to determine the K-nearest neighbors. The data for KNN algorithm consist of several multivariate attributes name that will be used to classify.

k' in KNN is a parameter that refers to the number of nearest neighbors to include in the majority of the voting process.

```
: 1 from sklearn.neighbors import KNeighborsClassifier
  2 classifier= KNeighborsClassifier(n_neighbors=1)
  3 classifier.fit(X_train, Y_train)
```

```
: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                       metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                       weights='uniform')
```

```
: 1 Y_pred= classifier.predict(X_test)
```

```
: 1 #Creating the Confusion matrix
  2 from sklearn.metrics import confusion_matrix,accuracy_score
  3 cm= confusion_matrix(Y_test, Y_pred)
```

```
: 1 print("Confusion Matrix \n",cm)
```

```
Confusion Matrix
 [[22870 15587]
 [ 3532 35168]]
```

```
: 1 acc = accuracy_score(Y_test,Y_pred)
```

```
: 1 print("Accuracy : ",acc)
```

```
Accuracy :  0.7522065399121273
```

4. Linear SVC

The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is. This makes this specific algorithm rather suitable for our uses, though you can use this for many situations.

```
: 1 from sklearn.svm import SVC # SVR
```

```
: 1 model = SVC()
  2 model.fit(X_train,Y_train)
  3 Y_predict=model.predict(X_test)
```

```
: 1 from sklearn.metrics import confusion_matrix,accuracy_score
  2 # Calculate Confusion Matrix
  3 cf_mat=confusion_matrix(Y_test,Y_predict)
```

```
: 1 acc = accuracy_score(Y_test,Y_predict)
```

```
: 1 print("Confusion Matrix \n",cf_mat)
  2 print("Accuracy : ",acc)
```

```
Confusion Matrix
 [[66305     0     0     0]
 [ 3902     0     0     0]
 [ 3877     0     0     0]
 [ 3874     0     0     0]]
Accuracy :  0.8505220759896355
```

5. Random Forest Model

Random Forest is an ensemble learning based regression model. It uses a model called decision tree, specifically as the name suggests, multiple decision trees to generate the ensemble model which collectively produces a prediction. The benefit of this model is that the trees are produced in parallel and are relatively uncorrelated, thus producing good results as each tree is not prone to individual errors of other trees.

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
param_grid = {
    'n_estimators': [20,50,100],
    }
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(random_state=7)

```

```
from sklearn.model_selection import GridSearchCV
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid,\
                      cv= 2, verbose=2)
```

```
CV_rfc.fit(X_train, Y_train)

```

]:
```
print(CV_rfc.best_params_)
print(CV_rfc.cv_results_['mean_test_score'])
print(sum(CV_rfc.cv_results_['mean_test_score'])/ len(CV_rfc.cv_results_['mean_test_score']))
```

```
{'n_estimators': 100}
[0.89812979 0.90014608 0.90092927]
0.8997350469772689
```

]:
```
rfc = RandomForestClassifier(n_estimators=100,random_state=7)
```

]:
```
rfc.fit(X_train,Y_train)

```

]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=7, verbose=0,
                       warm_start=False)
```

]:
```
Y_pred = rfc.predict(X_train)

```

]:
```
cf_mat=confusion_matrix(Y_train,Y_pred)
```

]:
```
acc = accuracy_score(Y_train,Y_pred)
```

]:
```
print("Confusion Matrix \n",cf_mat)
print("Accuracy : ",acc)
```

```
Confusion Matrix
 [[88802  1336]
 [ 2383 87512]]
Accuracy :  0.9793426760649436
```

6.Optimal Model

We can see among all these models Random Forest model has best fitted and givesbest Accuracy (R2) Score.

Algorithm	Score
Random Forest	0.9793
SVM	0.8505
KNN	0.7522
Logistic Regression	0.7522

7. Result and Module Accuracy Plot

We create classification report of our used models and plot the graph on the basis of models and their accuracy

```
Y_true = np.array(Y_test)
Y_pred = np.squeeze(np.array(rfc.predict(X_test) >= 0.5, dtype=np.int))
```

```
from sklearn.metrics import classification_report
```

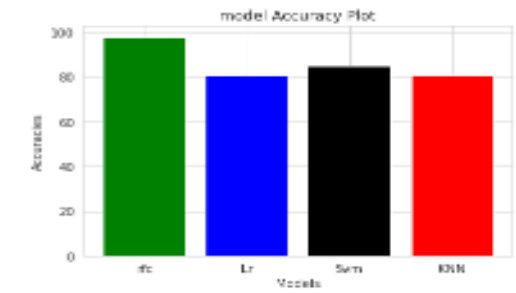
```
print(classification_report(Y_true, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.93      0.91      3236
           1       0.93      0.88      0.91      3343

    accuracy                           0.91      6579
   macro avg       0.91      0.91      0.91      6579
weighted avg       0.91      0.91      0.91      6579
```

```
x=['rfc','Lr','Svm','KNN']
y = [97.93,80.71,85.05,80.71]
```

```
['rfc', 'Lr', 'Svm', 'KNN']
```

```
plt.bar(x,y,color=['green', 'blue', 'black', 'red'])
plt.xlabel('Models')
plt.ylabel("Accuracies")
plt.title('model Accuracy Plot')
plt.show()
```



4.Requirements Specification

4.1 Hardware Requirement:

- 500 GB hard drive (Minimum requirement)
- 8 GB RAM (Minimum requirement)
- PC x64-bit CPU

4.2 Software Requirement:

- Windows/Mac/Linux
- Anaconda Navigator:
- Jupyter Notebook
    - **libraries :**
        - Pandas
        - Numpy
        - Matplotlib
        - Seaborn
        - Pickle

5.Conclusion:

By performing different ML models, we aim to get a better result or less errorwith max accuracy. Our purpose was to predict the age, height, weight, year, and season would affect on medal based on these few attributes. Initially, data cleaning is performed to remove the String values from the columns and converting it into integer values in the dataset. Next, with the help of data visualization features were explored deeply. The relation between the features is examined. Then ML models are implemented to predict the medal rate.

6.Future Scope

The paper Analyzing Sports Training Data with Machine Learning Techniques by Purdue University students, dove deeper in the machine learning aspect to improve the training and coaching of their Women's Soccer Team . Their data pre-processing included making the data anonymous and rearranging the data according to players vs according to training drills. There was player data corresponding to unique individuals and drill; data that was average out across all players. Furthermore, they normalized features into a common range.

7.References

Rgriffin. “120 Years of Olympic History: Athletes and Results.” *Kaggle* , 15 June 2018, www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results .

Jake Haas, Manuel Herrera *“Olympic History: Athletes and Results Data Analysis”* Computer Science,California State University, Sacramento Sacramento, CA, USA

VanderPlas, Jake. “Visualization with Seaborn.” *Visualization with Seaborn | Python Data Science Handbook* , jakevdp.github.io/PythonDataScienceHandbook/04.14-visualization-with-seaborn.html

Mahfuz, Rehana, et al. “[PDF] Analyzing Sports Training Data with Machine Learning Techniques: Semantic Scholar.” *Undefined* , 1 Jan. 1970, www.semanticscholar.org/paper/Analyzing-Sports-Training-Datawith-Machine-Mahfuz-Mourad/8a7a774e2aa3410575e0137071ed591fd65d1f78 .