Data Science amb Python

Cristiane de Souza da Silva

Sprint 6 : Sampling

May 2021

Exercises 1

Grab a sports-themed dataset you like. Performs a sampling of the data generating a simple random sample and a systematic sample.

Dataset

The selected dataset contains information about the modern Olympic Games, including all the Games from Athens 1896 to Rio 2016.

Attribute Information:

1. ID - Unique number for each athlete
2. Sex - M or F
3. Age - Integer
4. Height - In centimeters
5. Weight - In kilograms
6. NOC - National Olympic Committee 3-letter code
7. Season - Summer or Winter
8. Medal - Gold, Silver, Bronze, or NA

In [1]:
```
# import libraries needed

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#Load dataset
from datetime import datetime



games = pd.read_csv('athlete_olympics.csv')

games.head()
```

Out[2]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	1	M	24.0	180.0	80.0	CHN	Summer	NaN
1	2	M	23.0	170.0	60.0	CHN	Summer	NaN
2	3	M	24.0	NaN	NaN	DEN	Summer	NaN
3	4	M	34.0	NaN	NaN	DEN	Summer	Gold
4	5	F	21.0	185.0	82.0	NED	Winter	NaN

In [3]:
```
# show summary of Dataframe structure

games.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 8 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   ID      271116 non-null  int64  
 1   Sex     271116 non-null  object 
 2   Age     261642 non-null  float64
 3   Height  210945 non-null  float64
 4   Weight  208241 non-null  float64
 5   NOC     271116 non-null  object 
 6   Season  271116 non-null  object 
 7   Medal   39783 non-null   object 
dtypes: float64(3), int64(1), object(4)
memory usage: 16.5+ MB
```

In [4]:
```
# Summary statistics - Numeric variables (default)

games.describe()
```

Out[4]:

	ID	Age	Height	Weight
count	271116.000000	261642.000000	210945.000000	208241.000000
mean	68248.954396	25.556898	175.338970	70.702393
std	39022.286345	6.393561	10.518462	14.348020
min	1.000000	10.000000	127.000000	25.000000
25%	34643.000000	21.000000	168.000000	60.000000
50%	68205.000000	24.000000	175.000000	70.000000
75%	102097.250000	28.000000	183.000000	79.000000
max	135571.000000	97.000000	226.000000	214.000000

In [5]:
```
games.isnull().sum()
```

Out[5]:
```
ID             0
Sex            0
Age         9474
Height     60171
Weight     62875
NOC            0
Season         0
Medal     231333
dtype: int64
```

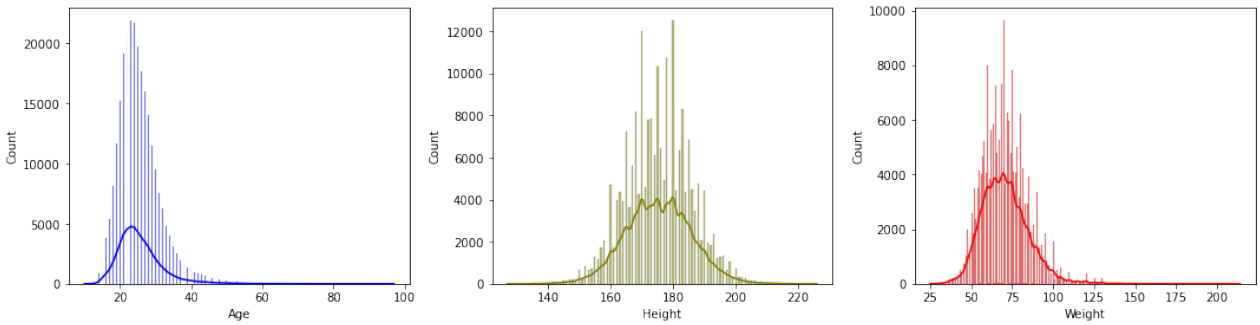
In [6]:
```
# Histogram for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.histplot(data=games, x="Age", kde=True, color="blue", ax=ax1)
sns.histplot(data=games, x="Height", kde=True, color="olive", ax=ax2)
sns.histplot(data=games, x="Weight", kde=True, color="red", ax=ax3)

plt.tight_layout()

plt.show()
```



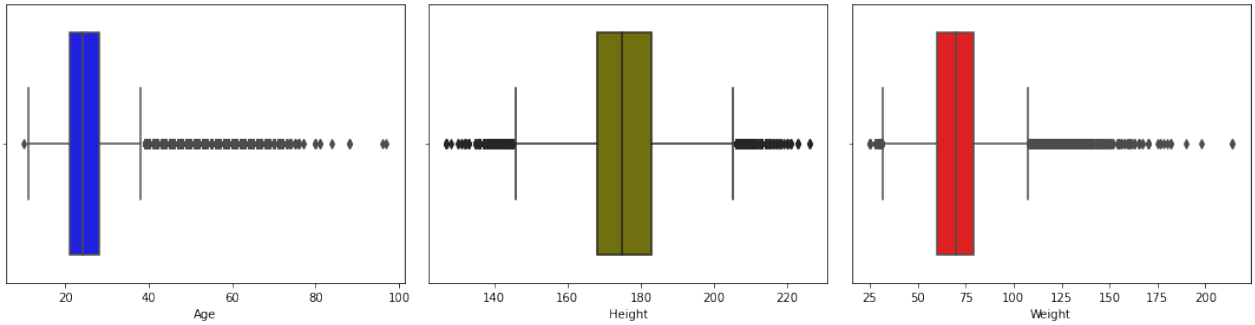
In [7]:
```
# Boxplot for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.boxplot(data=games, x="Age", color="blue", ax=ax1)
sns.boxplot(data=games, x="Height", color="olive", ax=ax2)
sns.boxplot(data=games, x="Weight", color="red", ax=ax3)

plt.tight_layout()

plt.show()
```



Simple random sample

Collect a simple random subset of games of 1000 observations.

In [8]:
```
# Plotting Random Sample Population

import random
random.seed(42)

k = 1000
sr_sample = games.sample(k).reset_index(drop=True)
sr_sample.head()
```

Out[8]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	96697	F	51.0	169.0	47.0	CAN	Summer	Bronze
1	98837	M	18.0	172.0	51.0	USA	Summer	Gold
2	47066	F	20.0	NaN	NaN	SWE	Summer	NaN
3	28519	M	31.0	188.0	NaN	IND	Summer	NaN
4	61986	M	27.0	171.0	70.0	GDR	Winter	Gold

In [9]:
```
sr_sample.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      1000 non-null   int64  
 1   Sex     1000 non-null   object 
 2   Age     963 non-null    float64
 3   Height  778 non-null    float64
 4   Weight  760 non-null    float64
 5   NOC     1000 non-null   object 
 6   Season  1000 non-null   object 
 7   Medal   167 non-null    object 
dtypes: float64(3), int64(1), object(4)
memory usage: 62.6+ KB
```

In [10]:
```
sr_sample.describe()
```

Out[10]:

	ID	Age	Height	Weight
count	1000.000000	963.000000	778.000000	760.000000
mean	68903.604000	25.574247	175.583548	71.442105
std	38416.416496	6.385033	10.273172	15.444360
min	19.000000	14.000000	144.000000	33.000000
25%	35435.750000	22.000000	170.000000	61.000000
50%	68335.500000	24.000000	175.000000	70.000000
75%	101970.500000	28.000000	183.000000	80.000000
max	135394.000000	65.000000	218.000000	182.000000

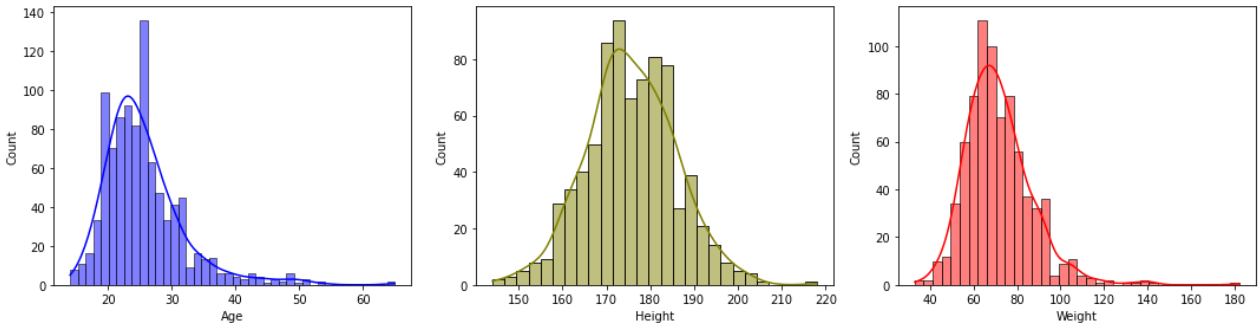
In [11]:
```
# Histogram for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.histplot(data=sr_sample, x="Age", kde=True, color="blue", ax=ax1)
sns.histplot(data=sr_sample, x="Height", kde=True, color="olive", ax=ax2)
sns.histplot(data=sr_sample, x="Weight", kde=True, color="red", ax=ax3)

plt.tight_layout()

plt.show()
```



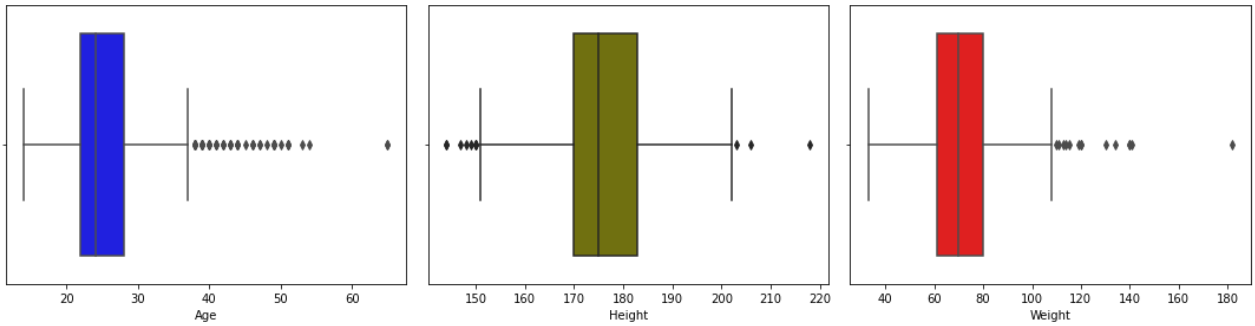
In [12]:
```
# Boxplot for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.boxplot(data=sr_sample, x="Age", color="blue", ax=ax1)
sns.boxplot(data=sr_sample, x="Height", color="olive", ax=ax2)
sns.boxplot(data=sr_sample, x="Weight", color="red", ax=ax3)

plt.tight_layout()

plt.show()
```



Systematic sample

Systematic sampling should not be used if the population is ordered cyclically or periodically, as the resulting sample cannot be guaranteed to be representative.

My list contains the entire population and is not in a periodic or cyclic order.

This sample will have also 55000 observations. To adapt the selection to the size of the Dataframe ('games'), the interval is calculated in the following way:

In [13]:
```
games.shape[0]
```

Out[13]: 271116

In [14]:
```
# Calculate the interval

# games.shape[0] = Size of population
# k = Size of sample

interval = int(games.shape[0]/1000 )
interval
```

Out[14]: 271

In [15]:
```
# Create function systematic sample

def sys_sample(df, size):

    '''
    This fuction returns the dataframe based on the DataFrame (population)
    and the sample size required
    '''


    indices = np.arange(0, len(df), step = df.shape[0]/size)

    sys_sample = df.iloc[indices]


    return sys_sample.reset_index(drop=True)
```

In [16]:
```
games_systematic = sys_sample(games, 1000)
games_systematic.head()
```

Out[16]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	1	M	24.0	180.0	80.0	CHN	Summer	NaN
1	141	F	30.0	167.0	58.0	IRL	Summer	NaN
2	311	M	21.0	185.0	80.0	IRQ	Summer	NaN
3	465	M	30.0	197.0	92.0	AUS	Summer	NaN
4	610	F	22.0	148.0	46.0	JPN	Summer	NaN

In [17]:
```
games_systematic['Sex'].value_counts()
```

Out[17]:
```
M    715
F    285
Name: Sex, dtype: int64
```

In [18]:
```
games_systematic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      1000 non-null   int64  
 1   Sex     1000 non-null   object 
 2   Age     967 non-null    float64
 3   Height  789 non-null    float64
 4   Weight  777 non-null    float64
 5   NOC     1000 non-null   object 
 6   Season  1000 non-null   object 
 7   Medal   132 non-null    object 
dtypes: float64(3), int64(1), object(4)
memory usage: 62.6+ KB
```

In [19]:
```
games_systematic.describe()
```

Out[19]:

	ID	Age	Height	Weight
count	1000.000000	967.000000	789.000000	777.000000
mean	68181.042000	25.739400	175.558935	71.197555
std	39042.739545	6.617338	10.519773	14.106801
min	1.000000	14.000000	142.000000	37.000000
25%	34606.750000	21.000000	168.000000	61.000000
50%	68137.000000	25.000000	176.000000	70.000000
75%	102005.750000	29.000000	183.000000	80.000000
max	135456.000000	65.000000	208.000000	130.000000

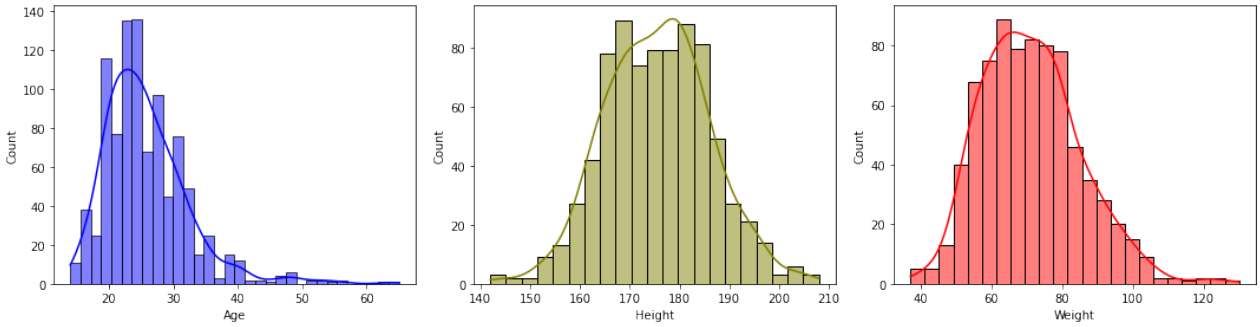
In [20]:
```
# Histogram for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.histplot(data=games_systematic, x="Age", kde=True, color="blue", ax=ax
sns.histplot(data=games_systematic, x="Height", kde=True, color="olive", a
sns.histplot(data=games_systematic, x="Weight", kde=True, color="red", ax=

plt.tight_layout()

plt.show()
```



Exercises 2

It continues with the sports theme data set and generates a stratified sample and a sample using SMOTE (Synthetic Minority Oversampling Technique).

Stratified sample

In [21]:
```
games_stratified = games.groupby('NOC', group_keys = False).apply(pd.DataF
games_stratified.head()
```

Out[21]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	99501	F	23.0	168.0	64.0	ALG	Summer	NaN
1	104800	M	23.0	175.0	73.0	ALG	Summer	NaN
2	93410	M	24.0	171.0	68.0	AND	Summer	NaN
3	18289	F	22.0	175.0	69.0	ANG	Summer	NaN
4	68692	M	24.0	160.0	60.0	ARG	Summer	NaN

In [22]:
```
games_stratified.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 997 entries, 0 to 996
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   ID      997 non-null    int64  
 1   Sex     997 non-null    object 
 2   Age     958 non-null    float64
 3   Height  779 non-null    float64
 4   Weight  776 non-null    float64
 5   NOC     997 non-null    object 
 6   Season  997 non-null    object 
 7   Medal   163 non-null    object 
dtypes: float64(3), int64(1), object(4)
memory usage: 62.4+ KB
```

In [23]:
```
games_stratified.describe()
```

Out[23]:

	ID	Age	Height	Weight
count	997.000000	958.000000	779.000000	776.000000
mean	70453.960883	25.765136	175.214377	71.161082
std	39203.136501	6.256295	10.300198	14.341992
min	6.000000	11.000000	138.000000	36.000000
25%	36878.000000	22.000000	168.000000	62.000000
50%	69882.000000	25.000000	175.000000	70.000000
75%	105057.000000	28.000000	182.000000	79.125000
max	135244.000000	58.000000	215.000000	141.000000

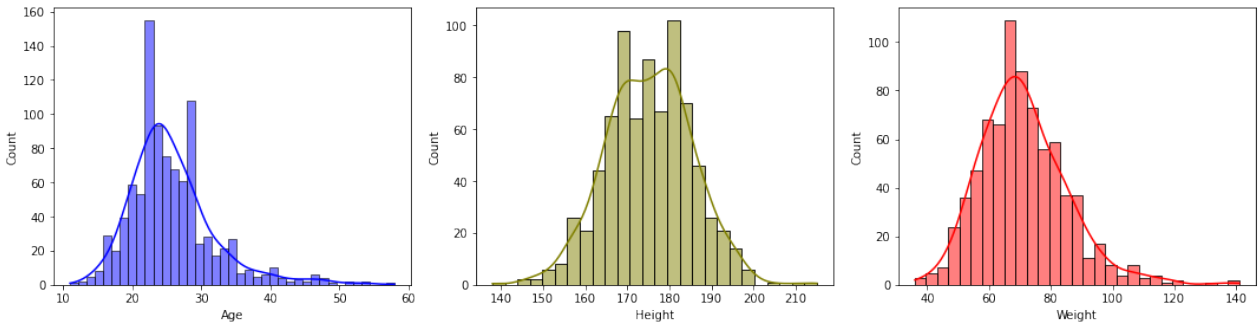
In [24]:
```
# Histogram for Age , Height and Weight


fig, (ax1, ax2, ax3) = plt.subplots(figsize=(15, 4), ncols=3)

sns.histplot(data=games_stratified, x="Age", kde=True, color="blue", ax=ax1
sns.histplot(data=games_stratified, x="Height", kde=True, color="olive", ax
sns.histplot(data=games_stratified, x="Weight", kde=True, color="red", ax=a

plt.tight_layout()

plt.show()
```



SMOTE

SMOTE (Syntetic Minority Oversampling Technique) means to synthesize elements for the minority class, based on the existing elements.

First, I'm going to copy the 'games' dataset and create the 'games_medal' dataset.

Afterwards, I will rank whether or not the person won a medal.

In [25]:
```
games_medal = games.copy()
games_medal.head()
```

Out[25]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	1	M	24.0	180.0	80.0	CHN	Summer	NaN
1	2	M	23.0	170.0	60.0	CHN	Summer	NaN
2	3	M	24.0	NaN	NaN	DEN	Summer	NaN
3	4	M	34.0	NaN	NaN	DEN	Summer	Gold
4	5	F	21.0	185.0	82.0	NED	Winter	NaN

In [26]:
```
games_medal['Sex'].value_counts()
```

Out[26]:
```
M    196594
F     74522
Name: Sex, dtype: int64
```

In [27]:
```
#Replace NaN Values with Zeros

games_medal['Medal'] = games_medal['Medal'].fillna(0)
games_medal.head()
```

Out[27]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	1	M	24.0	180.0	80.0	CHN	Summer	0
1	2	M	23.0	170.0	60.0	CHN	Summer	0
2	3	M	24.0	NaN	NaN	DEN	Summer	0
3	4	M	34.0	NaN	NaN	DEN	Summer	Gold
4	5	F	21.0	185.0	82.0	NED	Winter	0

In [28]:
```
games_medal['Sex'].value_counts()
```

Out[28]:
```
M    196594
F     74522
Name: Sex, dtype: int64
```

In [29]:
```
games_medal['Medal'].value_counts()
```

Out[29]:
```
0         231333
Gold       13372
Bronze     13295
Silver     13116
Name: Medal, dtype: int64
```

In [30]:
```
games_medal['Medal'] =games_medal['Medal'].replace(['Gold', 'Bronze', 'Silv
```

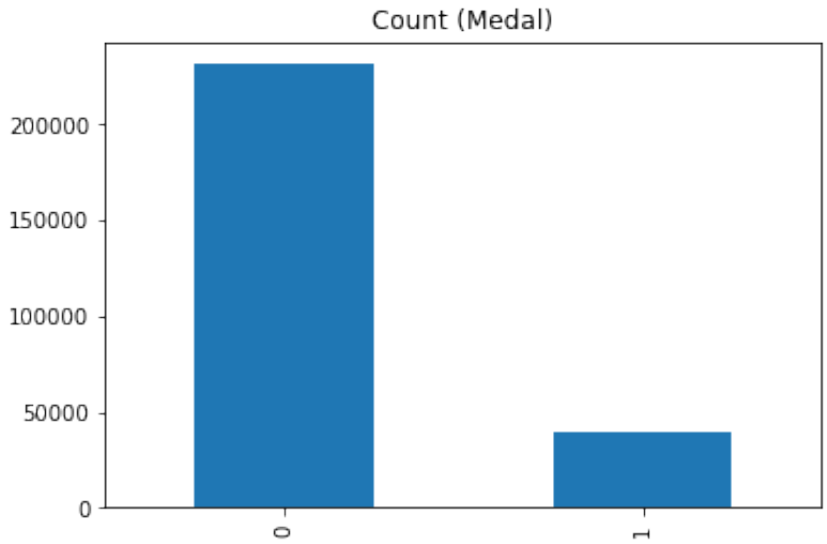
In [31]:
```
games_medal['Sex'].value_counts()
```

Out[31]:
```
M    196594
F     74522
Name: Sex, dtype: int64
```

In [32]:
```
games_medal['Medal'].value_counts()
```

Out[32]:
```
0    231333
1     39783
Name: Medal, dtype: int64
```

In [33]:
```
games_medal.Medal.value_counts().plot(kind='bar', title='Count (Medal)');
```



In [34]:
```
# Replace the Age, Height and Weight NaN by their median.

games_medal['Age'].fillna(games_medal['Age'].median(), inplace=True)

games_medal['Height'].fillna(games_medal['Height'].median(), inplace=True)

games_medal['Weight'].fillna(games_medal['Weight'].median(), inplace=True)
```

In [35]:
```
games_medal.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 8 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   ID      271116 non-null  int64  
 1   Sex     271116 non-null  object 
 2   Age     271116 non-null  float64
 3   Height  271116 non-null  float64
 4   Weight  271116 non-null  float64
 5   NOC     271116 non-null  object 
 6   Season  271116 non-null  object 
 7   Medal   271116 non-null  int64  
dtypes: float64(3), int64(2), object(3)
memory usage: 16.5+ MB
```

In [36]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

X = games_medal.drop(['Medal'], axis=1)
y = games_medal['Medal']
```

In [37]:
```
X.head()
```

Out[37]:

	ID	Sex	Age	Height	Weight	NOC	Season
0	1	M	24.0	180.0	80.0	CHN	Summer
1	2	M	23.0	170.0	60.0	CHN	Summer
2	3	M	24.0	175.0	70.0	DEN	Summer
3	4	M	34.0	175.0	70.0	DEN	Summer
4	5	F	21.0	185.0	82.0	NED	Winter

In [38]:
```
#convert text data to numeric before applying SMOTE
from sklearn import preprocessing

le = preprocessing.LabelEncoder()

X = X.apply(le.fit_transform)
```

In [39]:
```
X.head()
```

Out[39]:

	ID	Sex	Age	Height	Weight	NOC	Season
0	0	1	14	51	85	41	0
1	1	1	13	41	43	41	0
2	2	1	14	46	63	55	0
3	3	1	24	46	63	55	0
4	4	0	11	56	89	145	1

In [40]:
```
from imblearn.over_sampling import SMOTE

X_resampled, y_resampled = SMOTE(random_state=123).fit_resample(X, y)
```

In [41]:
```
y_resampled.value_counts()
```

Out[41]:
```
1    231333
0    231333
Name: Medal, dtype: int64
```

In [42]:
```
#Random over-sampling


ros_dataset = X_resampled
ros_dataset['Medal'] = y_resampled
ros_dataset.shape
```

Out[42]:
```
(462666, 8)
```

In [43]:
```
ros_dataset.head()
```

Out[43]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	0	1	14	51	85	41	0	0
1	1	1	13	41	43	41	0	0
2	2	1	14	46	63	55	0	0
3	3	1	24	46	63	55	0	1
4	4	0	11	56	89	145	1	0

Exercises 3

It continues with the sports theme dataset and generates a sample using the Reservoir sampling method.

In [44]:
```
# Resample the majority class without replacement

games_reservoir = games_medal.copy()
games_reservoir.head()
```

Out[44]:

	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	1	M	24.0	180.0	80.0	CHN	Summer	0
1	2	M	23.0	170.0	60.0	CHN	Summer	0
2	3	M	24.0	175.0	70.0	DEN	Summer	0
3	4	M	34.0	175.0	70.0	DEN	Summer	1
4	5	F	21.0	185.0	82.0	NED	Winter	0

Reservoir sampling is a family of randomized algorithms for choosing a simple random sample, without replacement.

So I choose the function **resample** from scikit-learn

In [45]:
```
from sklearn.utils import resample

games_reservoir_sampled = resample(games_reservoir,
                                   replace=False, # sample without replacer
                                   n_samples=1000, # number of sample
                                   random_state=123) # reproducible result
```

In [46]:
```
games_reservoir_sampled.reset_index()
```

Out[46]:

	index	ID	Sex	Age	Height	Weight	NOC	Season	Medal
0	217180	109088	M	26.0	178.0	83.0	GBR	Summer	0
1	23867	12532	M	29.0	176.0	70.0	GUA	Summer	0
2	50385	25925	M	25.0	175.0	70.0	BEL	Summer	0
3	137333	69041	F	17.0	165.0	57.0	FRA	Summer	0
4	9161	5042	M	30.0	166.0	62.0	PER	Summer	0
...	...	...	...	...	...	...	...	...	...
995	19130	10131	M	27.0	168.0	69.0	SUI	Summer	0
996	246522	123437	M	38.0	175.0	70.0	ITA	Summer	0
997	183550	92259	M	16.0	175.0	70.0	ITA	Summer	0
998	173693	87257	M	35.0	175.0	70.0	NOR	Summer	0
999	104580	52932	F	22.0	165.0	57.0	SCG	Summer	1

1000 rows × 9 columns

In [47]:
```
games_reservoir_sampled.shape
```

Out[47]:
```
(1000, 8)
```

In [ ]: