imporingt dependencies

```
#libaries or function that r required for this project
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn import metrics
```

```
#numpy is used to make arrays (list of numbers or values) and np is the abbreviation.
#pandas is used to make DataFrames (structured tables) makes it easy to analyze and process the data.
#(the files are inform of CSV (comma separated values).we can't analyze them like this so we convert them into dataframe using pandas)
#matplotlib is used to create plots(for data analysis we do some visualization) so it is for data visualization .
#seaborn is also for the data visualization.
#train_test_split is used to split the data we are importing it from sklearn .
#sklearn is one of the important machine learning libraries.
#XGBRgressor we are using this regressor in this project .
#metrices is used to evaluate the data. to know how well our model is working..
```

Data Collection and Data Processing

```
#loading data from csv file to Pandas DataFrame
calories = pd.read_csv('/content/calories.csv')
```

```
#print the first 5 rows of the DataFrame
calories.head()
```

| | User_ID | Calories |
|---|---|---|
| 0 | 14733363 | 231.0 |
| 1 | 14861698 | 66.0 |
| 2 | 11179863 | 26.0 |
| 3 | 16180408 | 71.0 |
| 4 | 17771927 | 35.0 |

```
exercise_data = pd.read_csv('/content/exercise.csv')
```

```
exercise_data.head()
```

| | User_ID | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp |
|---|---|---|---|---|---|---|---|---|
| 0 | 14733363 | male | 68 | 190.0 | 94.0 | 29.0 | 105.0 | 40.8 |
| 1 | 14861698 | female | 20 | 166.0 | 60.0 | 14.0 | 94.0 | 40.3 |
| 2 | 11179863 | male | 69 | 179.0 | 79.0 | 5.0 | 88.0 | 38.7 |
| 3 | 16180408 | female | 34 | 179.0 | 71.0 | 13.0 | 100.0 | 40.5 |
| 4 | 17771927 | female | 27 | 154.0 | 58.0 | 10.0 | 81.0 | 39.8 |

Combining two DataFrames

```
#combining the Calories col from calories dataframe into exercise_data dataframe.
#the calories burnt will be dependent of the Hear_Rate of an individual.
calories_data = pd.concat([exercise_data, calories['Calories']],axis=1)  #axis=1 is for col and 0 is for rows
```

```
calories_data.head()
```

| | User_ID | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp | Calories |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 14733363 | male | 68 | 190.0 | 94.0 | 29.0 | 105.0 | 40.8 | 231.0 |
| 1 | 14861698 | female | 20 | 166.0 | 60.0 | 14.0 | 94.0 | 40.3 | 66.0 |
| 2 | 11179863 | male | 69 | 179.0 | 79.0 | 5.0 | 88.0 | 38.7 | 26.0 |
| 3 | 16180408 | female | 34 | 179.0 | 71.0 | 13.0 | 100.0 | 40.5 | 71.0 |
| 4 | 17771927 | female | 27 | 154.0 | 58.0 | 10.0 | 81.0 | 39.8 | 35.0 |

```
#cheacking the no.of rows nd col

calories_data.shape
```

```
(15000, 9)
```

```
#getting some information about the data (useful for checking missing values)

calories_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   User_ID     15000 non-null  int64
 1   Gender      15000 non-null  object
 2   Age         15000 non-null  int64
 3   Height      15000 non-null  float64
 4   Weight      15000 non-null  float64
 5   Duration    15000 non-null  float64
 6   Heart_Rate  15000 non-null  float64
 7   Body_Temp   15000 non-null  float64
 8   Calories    15000 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 1.0+ MB
```

```
#cheacking the  missing values

calories_data.isnull().sum()
```

```
User_ID       0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
Body_Temp     0
Calories      0
dtype: int64
```

```
#if there are any missing values we need to preprocess the data .
#AND HERE WE DON'T HAVE ANY NULL VALUES.
```

## Data Analysis

```
#get some statical measurement of the data
calories_data.describe()
```

| | User_ID | Age | Height | Weight | Duration | Heart_Ra |
|---|---|---|---|---|---|---|
| **count** | 1.500000e+04 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.000000 | 15000.0000 |
| **mean** | 1.497736e+07 | 42.789800 | 174.465133 | 74.966867 | 15.530600 | 95.5185 |
| **std** | 2.872851e+06 | 16.980264 | 14.258114 | 15.035657 | 8.319203 | 9.5833 |
| **min** | 1.000116e+07 | 20.000000 | 123.000000 | 36.000000 | 1.000000 | 67.0000 |
| **25%** | 1.247419e+07 | 28.000000 | 164.000000 | 63.000000 | 8.000000 | 88.0000 |
| **50%** | 1.499728e+07 | 39.000000 | 175.000000 | 74.000000 | 16.000000 | 96.0000 |
| **75%** | 1.744928e+07 | 56.000000 | 185.000000 | 87.000000 | 23.000000 | 103.0000 |
| **max** | 1.999965e+07 | 79.000000 | 222.000000 | 132.000000 | 30.000000 | 128.0000 |

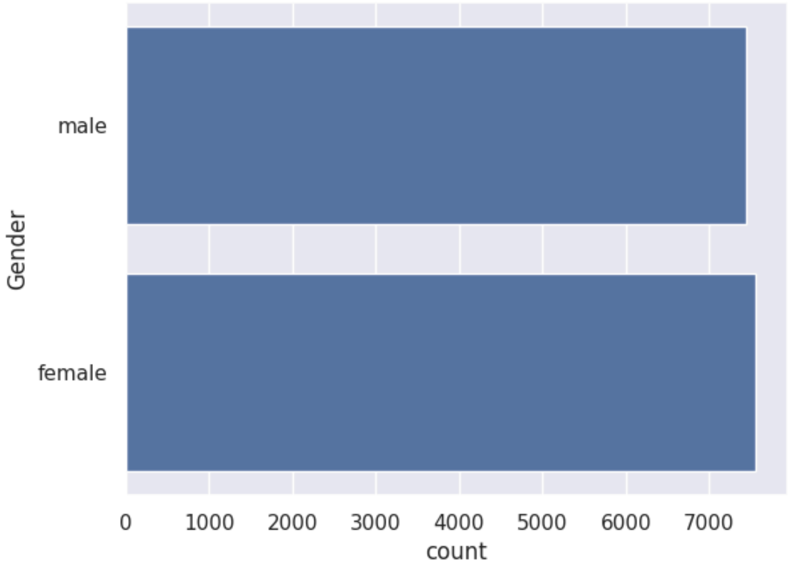
## Data Visualization

```
sns.set()
#gives baisc theme for the plots like grids and Background colour.
```

```
#Plotting the gender col in countplot

sns.countplot(calories_data['Gender'])
```

```
<Axes: xlabel='count', ylabel='Gender'>
```




```
#as the distribution of gender are equal it means that the data is very good distributed dataset.
#we can use countplot only for categorical cols like male or female, Yes or NO. can't useful for numerical cols.
#for numerical cols we use distplot for visualization of the data.
```

```
#finding the distribution of the 'AGE' col.
sns.distplot(calories_data['Age'])
```

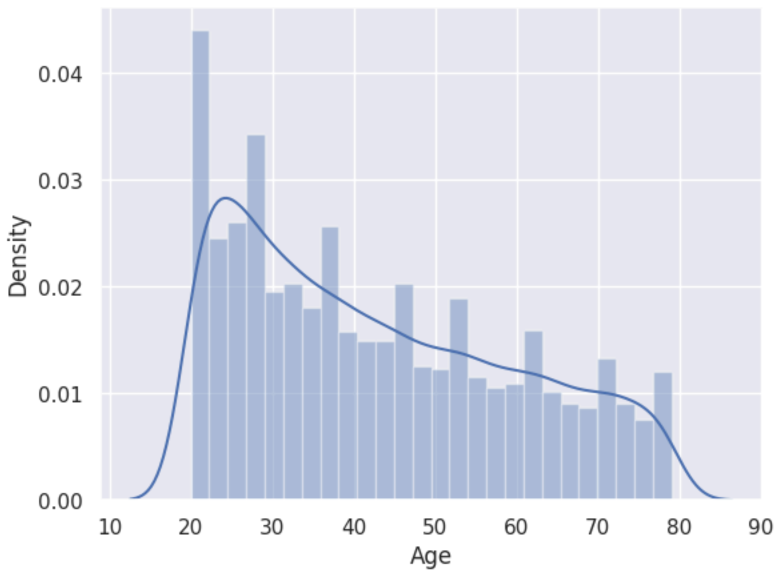
```
<ipython-input-17-d69e6e018a56>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Age'])
<Axes: xlabel='Age', ylabel='Density'>
```




```
#finding the distribution of the 'Height' col.
sns.distplot(calories_data['Height'])
```

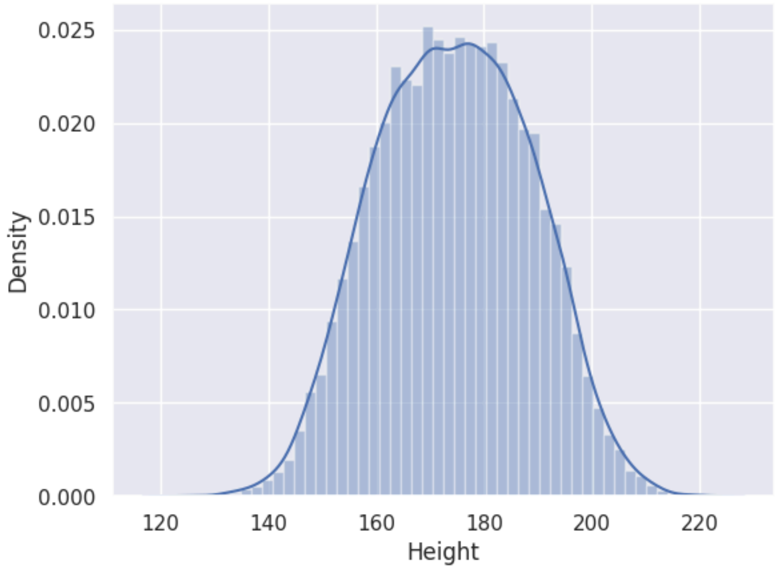
```
<ipython-input-18-8e8a5e46a286>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Height'])
<Axes: xlabel='Height', ylabel='Density'>
```




```
#finding the distribution of the 'Weight' col.
sns.distplot(calories_data['Weight'])
```

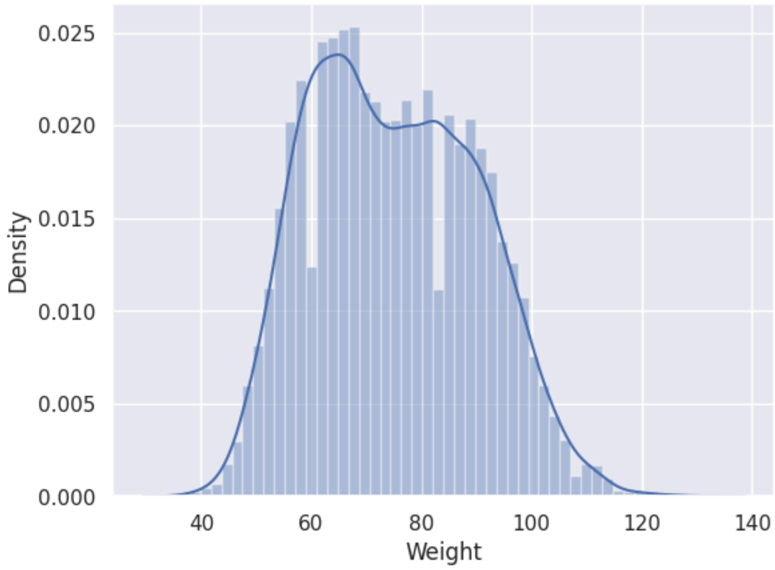
```
<ipython-input-19-5ad90a9eb752>:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(calories_data['Weight'])
<Axes: xlabel='Weight', ylabel='Density'>
```




Converting text data into numerical data

```
#we are coverting the text col of "GENDER" into numerical col becoz we can't feed a textdata to a ML model
#for that purpose we need to covert into numerical values.
```

```
#categorical convertion
```

```
calories_data["Gender"].replace({"male": 0, "female": 1}, inplace=True)
```

```
calories_data.head()
```

| | User_ID | Gender | Age | Height | Weight | Duration | Heart_Rate | Body_Temp | Calories |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 14733363 | 0 | 68 | 190.0 | 94.0 | 29.0 | 105.0 | 40.8 | 231.0 |
| **1** | 14861698 | 1 | 20 | 166.0 | 60.0 | 14.0 | 94.0 | 40.3 | 66.0 |
| **2** | 11179863 | 0 | 69 | 179.0 | 79.0 | 5.0 | 88.0 | 38.7 | 26.0 |
| **3** | 16180408 | 1 | 34 | 179.0 | 71.0 | 13.0 | 100.0 | 40.5 | 71.0 |
| **4** | 17771927 | 1 | 27 | 154.0 | 58.0 | 10.0 | 81.0 | 39.8 | 35.0 |

Finding the Correaltion in the Dataset

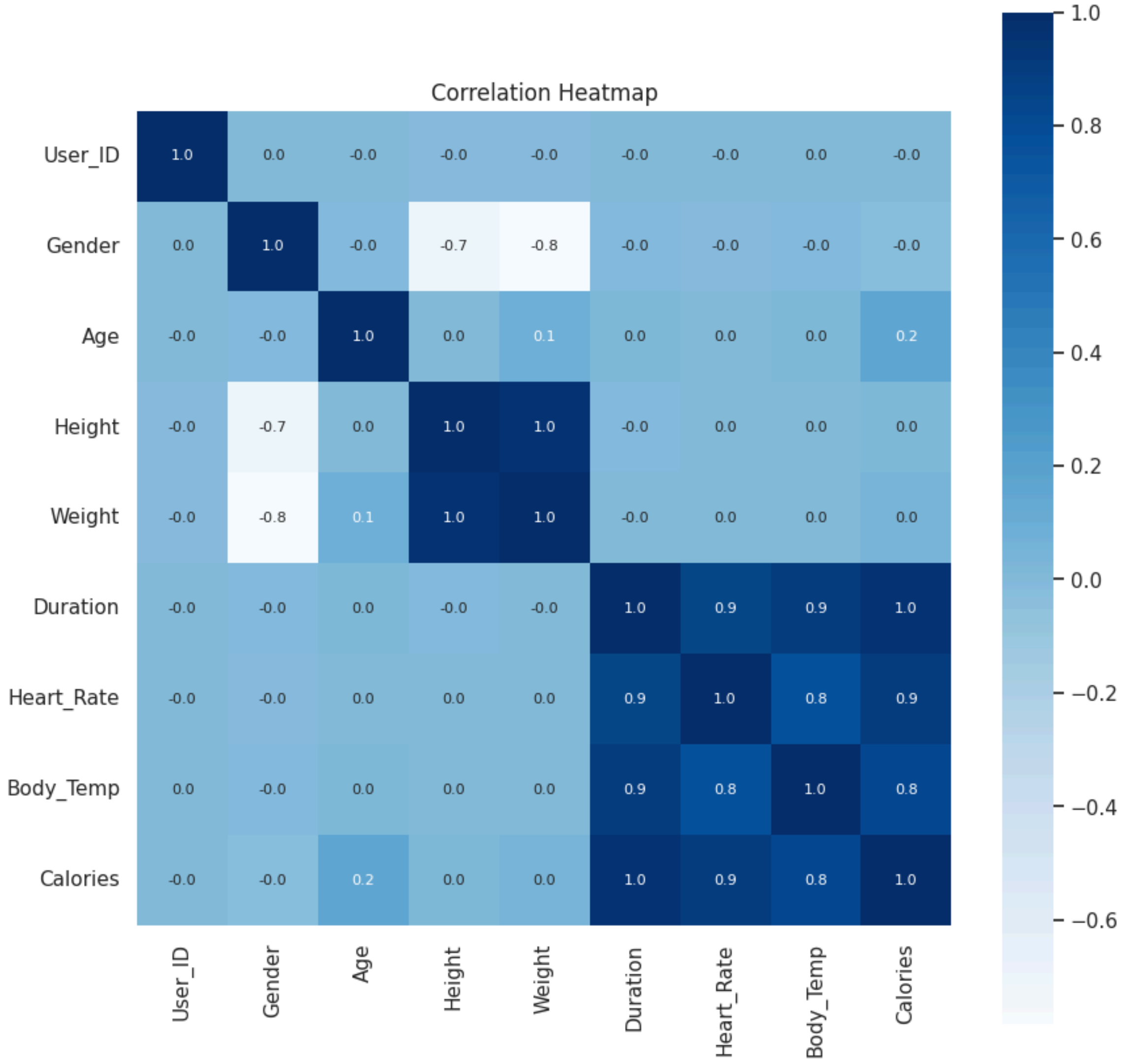
1.Positive Correlation. 2.Negative Correaltion.

```
#Positive corealtion: when 2 cols are directly proportional i.e
#example: when Duration increases the Calories Burnt are also increases.
#Negative correaltion: When they r indirectly proportional.
#AND "corr" function is used to calculate the correaltion
```

```
correlation = calories_data.corr()
```

```
#constructing a heatmat to understand the correlation

plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')
plt.title('Correlation Heatmap')
plt.show()
```



```
#the HEATMAP gives colors according to the values.
#each col is copared to other col if the value is large than is gives "1" and if it is less it gives "0"
#1.0 means that the relation btw them is "Positively Correlations".
#if the val is very less then it is "Negatively Correlations".
#if the val is 0 then there is no correlations.
#Height and Weight are compared and are positive are correlated  (if a person is v.height then that weight will be more).
#Duration,Heart_rate,Body_Temp and Calories are correlated (if a person spends more time then they will get increase in heart_rate, body
#burn more calories).
```

Separating Features and target

```
#dividing the dataset into two parts.
#we need to predict calories so taking into a diff Y variable.
#and rest of cols into X variable.
```

```
X= calories_data.drop(columns=['User_ID','Calories'], axis=1)
Y= calories_data['Calories']
```

```
print(X)
```

```
       Gender  Age  Height  Weight  Duration  Heart_Rate  Body_Temp
0           0   68   190.0    94.0      29.0       105.0       40.8
1           1   20   166.0    60.0      14.0        94.0       40.3
2           0   69   179.0    79.0       5.0        88.0       38.7
3           1   34   179.0    71.0      13.0       100.0       40.5
4           1   27   154.0    58.0      10.0        81.0       39.8
...       ...  ...     ...     ...       ...         ...        ...
14995       1   20   193.0    86.0      11.0        92.0       40.4
14996       1   27   165.0    65.0       6.0        85.0       39.2
14997       1   43   159.0    58.0      16.0        90.0       40.1
14998       0   78   193.0    97.0       2.0        84.0       38.3
14999       0   63   173.0    79.0      18.0        92.0       40.5

[15000 rows x 7 columns]
```

```
print(Y)
```

```
0        231.0
1         66.0
2         26.0
3         71.0
4         35.0
         ...
14995     45.0
14996     23.0
14997     75.0
14998     11.0
14999     98.0
Name: Calories, Length: 15000, dtype: float64
```

Splitting dataset into training and testing

```
#X_train consist all the traning data of X
#X_test contains all the tests data of X
#Y_train contain the corresponding calories of X_train are stored.
#Y_test contains the corresponding calories of X_train are stored.
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2,random_state=2)
```

```
#0.2 means 20% of testing data we are considering as testing data.
#80% as training data to train the model.
```

```
print(X.shape, X_train.shape, X_test.shape)
```

```
(15000, 7) (12000, 7) (3000, 7)
```

Modelling training

XGBoost Regression

```
#Loading the model
```

```
model = XGBRegressor()
```

```
#training the model with X_train

model.fit(X_train,Y_train)
```

```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

```
# this training model if given to XGBoost regression model it finds the pattern btw the models
#the model will automatically will learn that if duration is more the calories burnt will be more
#or if the heart_rate is more then the calories burnt will be more like that this will be understand by the model through this
#training data .   NOW THE MODEL HAS LEARNT FROM THE DATA.
```

Evaluation

predicting on testing data

```
test_data_prediction = model.predict (X_test)
```

```
#it will verfiy the corresponding calories burnt from the X_test features like:Duration,Heart_rate etc...
#save those calories in the test_data_prediction.
#Now compare this values with Y_test.
```

```
print(test_data_prediction)
```

```
[125.58828  222.11377   38.725952 ... 144.3179    23.425894  90.100494]
```

Mean Absolute Error

```
#Here we compare the original calories values with test_data_prediction .
#if we see the above there is 38.7259 and if the original is 33.873 then we will subtract the and we will take the mean
#so like this we will consider the overall values and will predict the MEAN Error.
```

```
mae=metrics.mean_absolute_error(Y_test,test_data_prediction)
```

```
print("Mean Absolute Error:",mae)
```

```
Mean Absolute Error: 1.4833678883314132
```

```
#As the mean is very low our model is good....
```

Buliding a Predictive System

```
input_data=(0,68,190.0,94.0,29.0,105.0,40.8)

#chaning input_data into numpy array
input_data_as_numpy_array = np.asarray(input_data)

#reshaping the array
input_data_reshape = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshape)
print(prediction)
```

```
[236.13371]
```

```
#we changing the input_Data into numpy array becoz it is easy to some processing on NUMPY rather than tuples

#we are using reshape to tell the model that we are considering  single row or data.

#the output we got is 236 which is near to the orginal val(231) so that means the model is working perfectly .
```