Importing the Dependencies

```
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
```

Data Collection & Processing

```
# loading the data from csv file to a 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 the two Dataframes

```
calories_data = pd.concat([exercise_data, calories['Calories']], axis=1)
```

```
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

```
# checking the number of rows and columns
calories_data.shape
```

```
(15000, 9)
```

```
# getting some informations about the data
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
```

```
# checking for 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
```

Data Analysis

```
# get some statistical measures about the data
calories_data.describe()
```

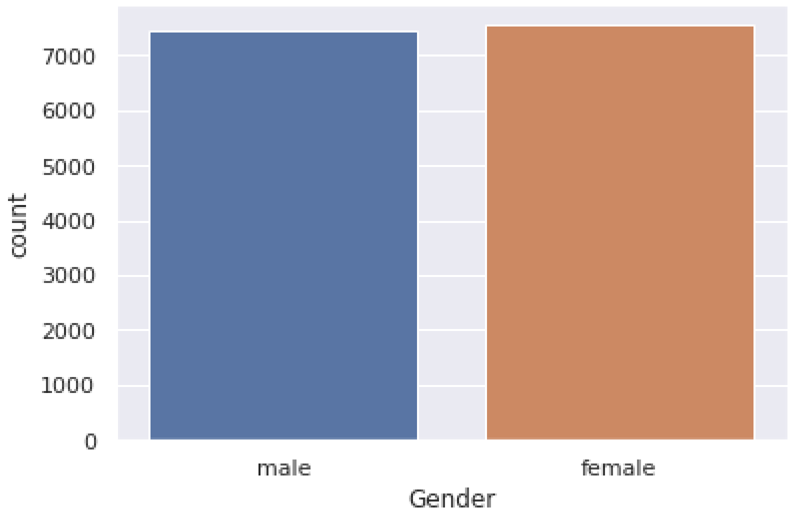
	User_ID	Age	Height	Weight	Duration	Heart_R
count	1.500000e+04	15000.000000	15000.000000	15000.000000	15000.000000	15000.000
mean	1.497736e+07	42.789800	174.465133	74.966867	15.530600	95.518
std	2.872851e+06	16.980264	14.258114	15.035657	8.319203	9.583
min	1.000116e+07	20.000000	123.000000	36.000000	1.000000	67.000
25%	1.247419e+07	28.000000	164.000000	63.000000	8.000000	88.000
50%	1.499728e+07	39.000000	175.000000	74.000000	16.000000	96.000
75%	1.744928e+07	56.000000	185.000000	87.000000	23.000000	103.000
max	1.999965e+07	79.000000	222.000000	132.000000	30.000000	128.000

Data Visualization

```
sns.set()
```

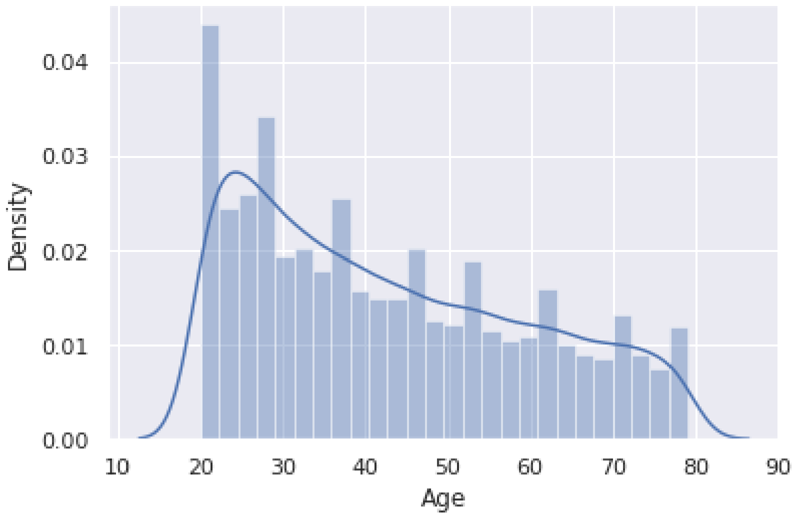
```
# plotting the gender column in count plot
sns.countplot(calories_data['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a23007110>
```



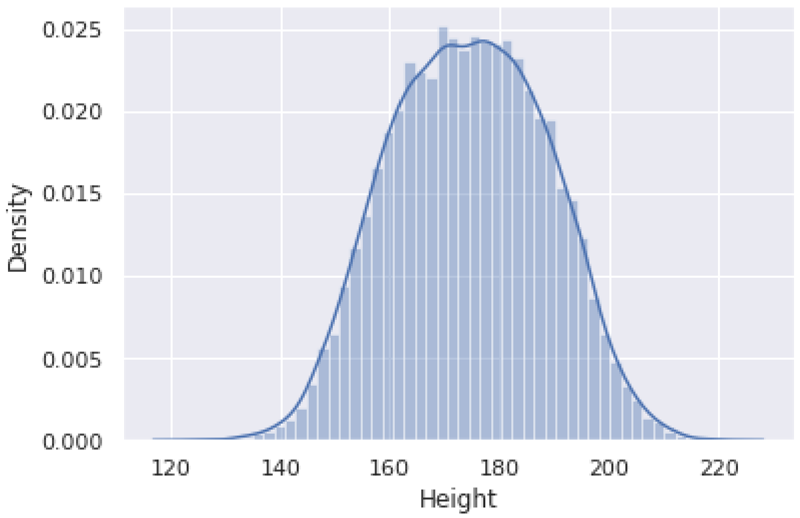
```
# finding the distribution of "Age" column
sns.distplot(calories_data['Age'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a22b17e50>
```



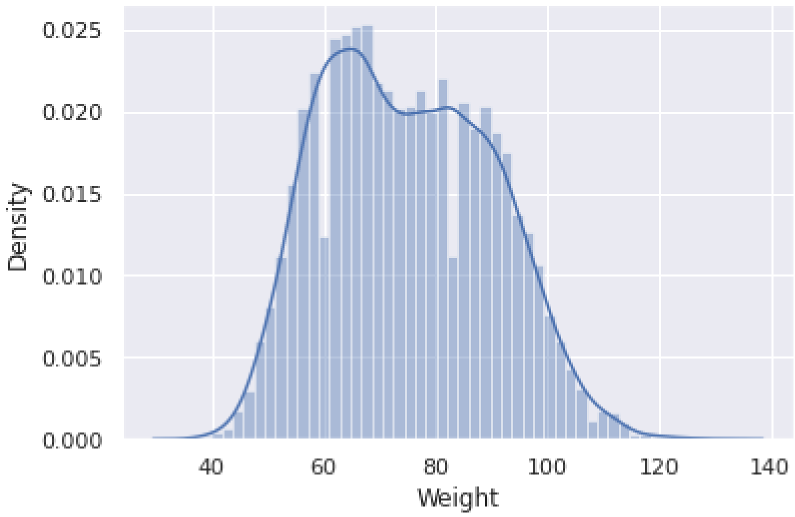

```
# finding the distribution of "Height" column
sns.distplot(calories_data['Height'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a17f6ec90>
```



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

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a17f6fad0>
```



Finding the Correlation in the dataset

1. Positive Correlation
2. Negative Correlation

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

```
# constructing a heatmap to understand the correlation

plt.figure(figsize=(10,10))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a17dbae90>
```

Converting the text data to numerical values

```
calories_data.replace({"Gender":{'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

Separating features and Target

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

```
print(X.head())
```

```
   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
```

```
print(Y.head())
```

```
0    231.0
1     66.0
2     26.0
3     71.0
4     35.0
Name: Calories, dtype: float64
```

Splitting the data into training data and Test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)
```

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

```
(15000, 7) (10500, 7) (4500, 7)
```

Model Training

XGBoost Regressor

```
# loading the model
model = XGBRegressor()
```

```
# training the model with X_train
model.fit(X_train, Y_train)
```

```
[11:47:16] WARNING: /workspace/src/objective/regression_obj.cu:152: reg:linear is no
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, gamma=0,
             importance_type='gain', learning_rate=0.1, max_delta_step=0,
             max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
             n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
```

```
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

Evaluation

Prediction on Test Data

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

```
print(test_data_prediction)
```

```
[167.63509  193.85378   51.93017  ...  30.419823 188.78845  148.03796 ]
```

Mean Absolute Error

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

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

```
Mean Absolute Error =  2.622601093477673
```

Score on test and train dataset

```
model.score(X_train, Y_train)
```

```
0.9969798435494149
```

```
model.score(X_test, Y_test)
```

```
0.9965459879538597
```