Calories Burnt Predictor

```
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 Collecting and Preprocessing

```
calories = pd.read_csv('calories.csv')
exercise = pd.read_csv('exercise.csv')
```

Combining Two dataframes

```
data = pd.concat([exercise, calories['Calories']], axis=1)
```

```
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

Next steps: Generate code with data View recommended plots New interactive sheet

```
data.shape
```

```
(15000, 9)
```

checking for missing values

```
data.isnull().sum()
```

	0
User_ID	0
Gender	0
Age	0
Height	0
Weight	0
Duration	0
Heart_Rate	0
Body_Temp	0
Calories	0

dtype: int64

What can I help you build?

Data Analysis

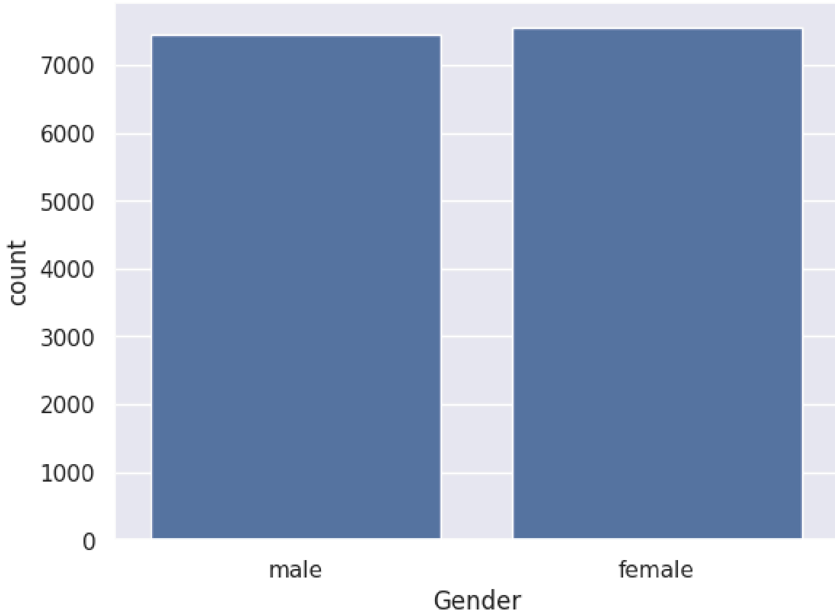
```
data.describe()
```

	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
count	1.500000e+04	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000	15000.000000
mean	1.497736e+07	42.789800	174.465133	74.966867	15.530600	95.518533	40.025453	89.539533
std	2.872851e+06	16.980264	14.258114	15.035657	8.319203	9.583328	0.779230	62.456978
min	1.000116e+07	20.000000	123.000000	36.000000	1.000000	67.000000	37.100000	1.000000
25%	1.247419e+07	28.000000	164.000000	63.000000	8.000000	88.000000	39.600000	35.000000
50%	1.499728e+07	39.000000	175.000000	74.000000	16.000000	96.000000	40.200000	79.000000
75%	1.744928e+07	56.000000	185.000000	87.000000	23.000000	103.000000	40.600000	138.000000
max	1.999965e+07	79.000000	222.000000	132.000000	30.000000	128.000000	41.500000	314.000000

```
sns.set()
```

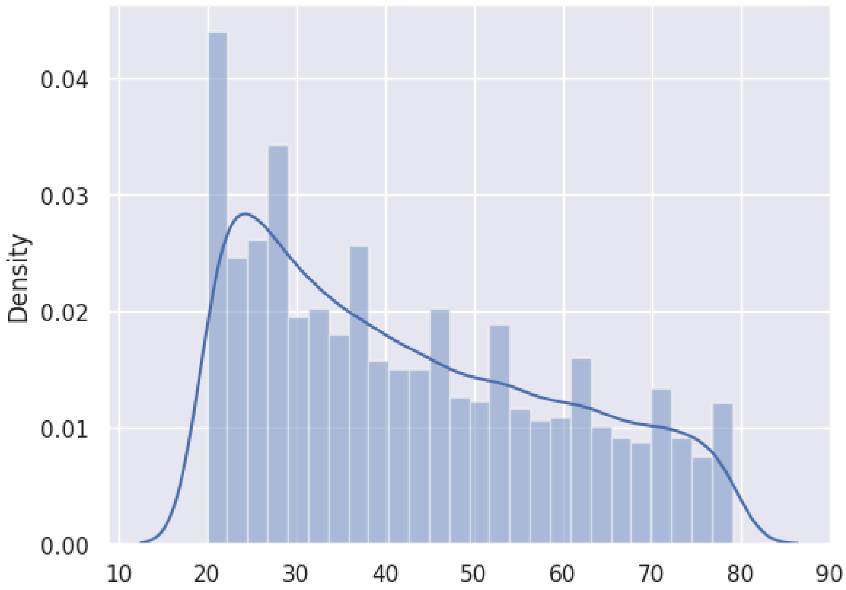
```
sns.countplot(x=data['Gender'])
```

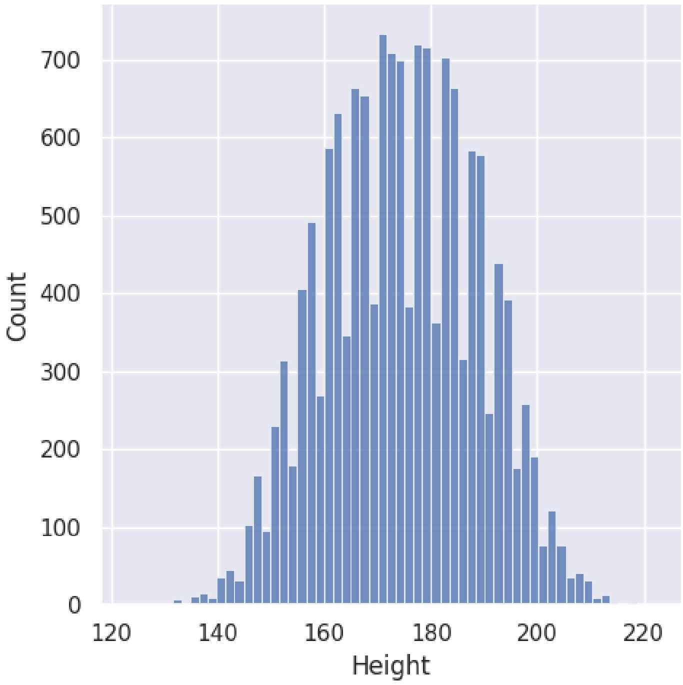
```
<Axes: xlabel='Gender', ylabel='count'>
```

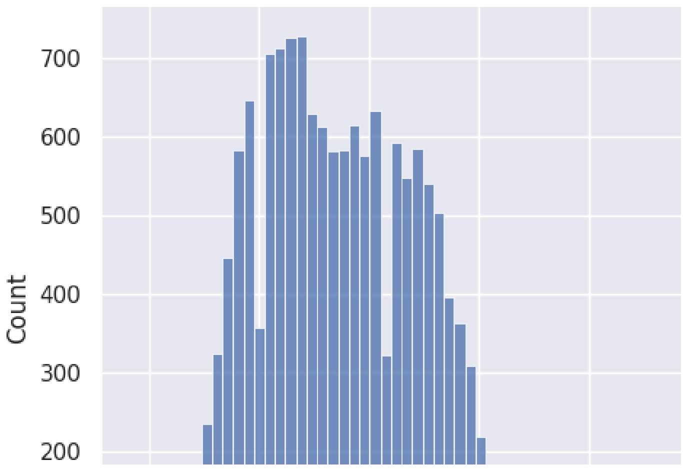


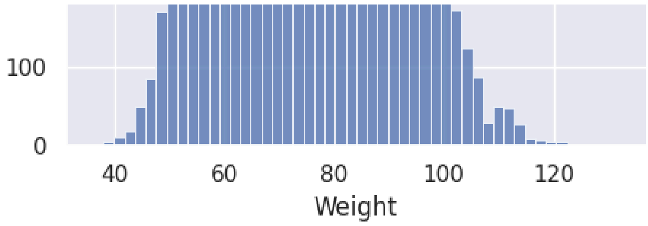

```
sns.distplot(x=data['Age'])
sns.displot(x=data['Height'])
sns.displot(x=data['Weight'])
```

```
/tmp/ipython-input-13-1846813509.py:1: 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(x=data['Age'])
<seaborn.axisgrid.FacetGrid at 0x7898a7e97ad0>
```









Label Encoding

```
data.replace({"Gender":{'male':0,'female':1}}, inplace=True)
data.head()
```

```
/tmp/ipython-input-15-4057041858.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and wil
  data.replace({"Gender":{'male':0,'female':1}}, inplace=True)
```

	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

Next steps: Generate code with data | View recommended plots | New interactive sheet

Splitting Data and target

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

splitting test and train data

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

Model Training(XGBoost Regressor)

```
model = XGBRegressor()
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, ...)
```

Model Evaluation

On Train data

```
train_data_Prediction = model.predict(X_train)
mae = metrics.mean_absolute_error(Y_train, train_data_Prediction)
mse = metrics.mean_squared_error(Y_train, train_data_Prediction)
print("Mean Absolute Error: ", mae)
print("Mean Squared Error: ", mse)
```

```
Mean Absolute Error:  0.9322033420062313
Mean Squared Error:  1.6776731334332036
```

On test data

```
test_data_Prediction = model.predict(X_test)
mae = metrics.mean_absolute_error(Y_test, test_data_Prediction)
mse = metrics.mean_squared_error(Y_test, test_data_Prediction)
print("Mean Absolute Error: ", mae)
print("Mean Squared Error: ", mse)
```

```
Mean Absolute Error:  1.4833678883314132
Mean Squared Error:  4.710710012461346
```

Building A Predictive System

```
input = (0,68,190.0,94.0,29.0,105.0,40.8)
input_as_numpy_array = np.asarray(input)
input_reshaped = input_as_numpy_array.reshape(1,-1)
prediction = model.predict(input_reshaped)
print(prediction)
```

```
[236.13371]
```

```
input =(1,34,179.0,71.0,13.0,100.0,40.5)
input_as_numpy_array = np.asarray(input)
```