In [2]:
```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [5]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler,OrdinalEncoder
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split,cross_val_score,KFold
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
```

In [14]:
```
def read_csv(file_path):
    return pd.read_csv(file_path)

def dataset_info_statistics(data):
    print(data.info())
    print("\n")

    print("Basic Statistics for numerical column:")
    print(data.describe())
    print("\n")
#check for null values in the datasets
def check_null(data):
    null_counts = data.isnull().sum()
    print("NUll Values in the Datasets:")
    return null_counts
#checks for duplicated rows in the dataset
def check_duplicate(data):
    """
    Parameters: Panda Dataframe, input data

    Returns: - bool,True if any duplicated rows exist, False otherwise.
    """
    return data.duplicated().any()
#Getting basic analysis for numerical and categorical columns
# 1. Reading data from CSV
def read_csv(file_path):
    """
    Read data from a CSV file and return a pandas DataFrame.

    Parameters:
    - file_path: str, the path to the CSV file.

    Returns:
    - pd.DataFrame, the loaded DataFrame.
    """
    return pd.read_csv(file_path)
#2. Getting information and statistics about over dataset
def dataset_info_statistics(data):
    """
    Display information and basic statistics about the dataset.

    Parameters:
    - data: pandas DataFrame, input data.

    Returns:
```

```
    - None
    """
    # Display general information about the dataset
    print("Dataset Information:")
    print(data.info())
    print("\n")

    # Display basic statistics for numerical columns
    print("Basic Statistics for Numerical Columns:")
    print(data.describe())
    print("\n")

#3.check for the null values in the dataset
def check_null(data):
    """
    Check for null values in the dataset.

    Parameters:
    - data: pandas DataFrame, input data.

    Returns:
    - pd.Series, the count of null values for each column.
    """
    null_counts = data.isnull().sum()
    print("Null Values in the Dataset:")
    return null_counts

#4.check for duplicated rows in the dataset
def check_duplicates(data):
    """
    Check for duplicated rows in the dataset.

    Parameters:
    - data: pandas DataFrame, input data.

    Returns:
    - bool, True if any duplicated rows exist, False otherwise.
    """
    return data.duplicated().any()

#5. getting basic analysis for numerical and categorical columns
def plot_graph(data):
    """
    Plot graphs for numerical and categorical data in a dataframe.

    Parameters:
    - data: Pandas Dataframe, input data.

    Returns:
    - None

    """
    numerical_columns = data.select_dtypes(include=np.number).columns

    for column in numerical_columns:
        plt.figure(figsize=(5,3))
        sns.distplot(data[column],kde=True)
        plt.title(f"Histogram for {column}")
        plt.xlabel(column)
        plt.ylabel("Frequency")
        plt.show()

    categorical_columns = data.select_dtypes(include='object').columns
    for column in categorical_columns:
```

```
        plt.figure(figsize=(5, 3))
        sns.countplot(data[column])
        plt.title(f'Countplot for {column}')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.xticks(rotation=45)
        plt.show()

def sep_feature_target(data,target_column):
    """
    Separate features and target variable

    Parameters :
     -data  - pandas dataframe, input data
     -target_column : str, the column representing the target variable

     returns - x- pandas dataframe, features
               y - pandas series, target variables
    """
    X = data.drop(columns=[target_column],axis=1)
    Y=data[target_column]
    return X,Y

def perform_train_test_split(x,y,test_size=0.20,random_state=42):
    """
    Performs train test split
    parameters :
    - x: pandas Dataframe, features
    - y: pandas Series, target variable
    -test_size :float, optional, the proporation of the dataset to include in the t
    -random_state : int, optional, seed fro random number generation (default is 42

    returns -
    - x_train : Pandas Dataframe, features for training
    - y_test :  Pandas Dataframe, features for testing
    - x_train:  Pandas series, target variable for training
    - y_test:  Pandas Series, target variable for testing
    """
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=test_size,random
    return x_train,x_test,y_train,y_test
```

In [7]:
```
c = read_csv('C:/Users/Windows/Downloads/calories.csv')
e = read_csv('C:/Users/Windows/Downloads/exercise.csv')
```

In [8]:
```
data = pd.merge(c,e,on='User_ID')
```

In [9]:
```
data.head()
```

Out[9]:

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

In [10]:
```
dataset_info_statistics(data)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15000 entries, 0 to 14999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   User_ID     15000 non-null  int64  
 1   Calories    15000 non-null  float64
 2   Gender      15000 non-null  object 
 3   Age         15000 non-null  int64  
 4   Height      15000 non-null  float64
 5   Weight      15000 non-null  float64
 6   Duration    15000 non-null  float64
 7   Heart_Rate  15000 non-null  float64
 8   Body_Temp   15000 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 1.1+ MB
None


Basic Statistics for numerical column:
            User_ID      Calories           Age        Height        Weight  \
count  1.500000e+04  15000.000000  15000.000000  15000.000000  15000.000000   
mean   1.497736e+07     89.539533     42.789800    174.465133     74.966867   
std    2.872851e+06     62.456978     16.980264     14.258114     15.035657   
min    1.000116e+07      1.000000     20.000000    123.000000     36.000000   
25%    1.247419e+07     35.000000     28.000000    164.000000     63.000000   
50%    1.499728e+07     79.000000     39.000000    175.000000     74.000000   
75%    1.744928e+07    138.000000     56.000000    185.000000     87.000000   
max    1.999965e+07    314.000000     79.000000    222.000000    132.000000   

            Duration    Heart_Rate     Body_Temp  
count  15000.000000  15000.000000  15000.000000  
mean      15.530600     95.518533     40.025453  
std        8.319203      9.583328      0.779230  
min        1.000000     67.000000     37.100000  
25%        8.000000     88.000000     39.600000  
50%       16.000000     96.000000     40.200000  
75%       23.000000    103.000000     40.600000  
max       30.000000    128.000000     41.500000  
```

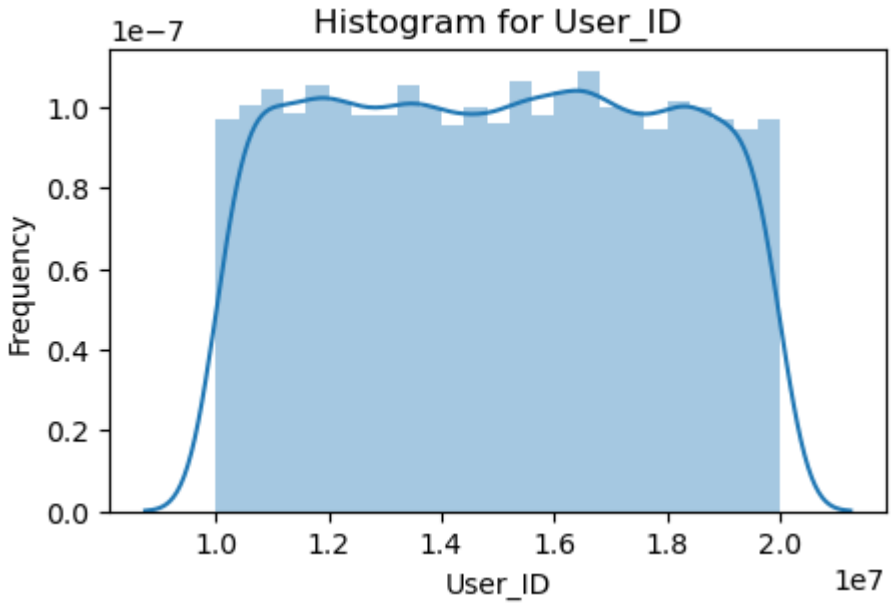
In [11]:
```
check_null(data)
```

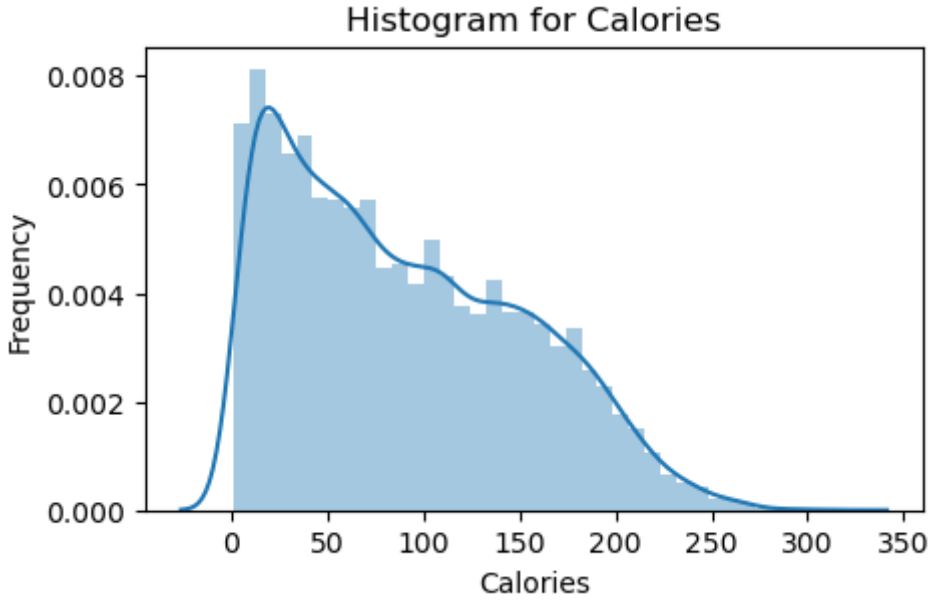
Out[11]:
```
NUll Values in the Datasets:
User_ID       0
Calories      0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
Body_Temp     0
dtype: int64
```

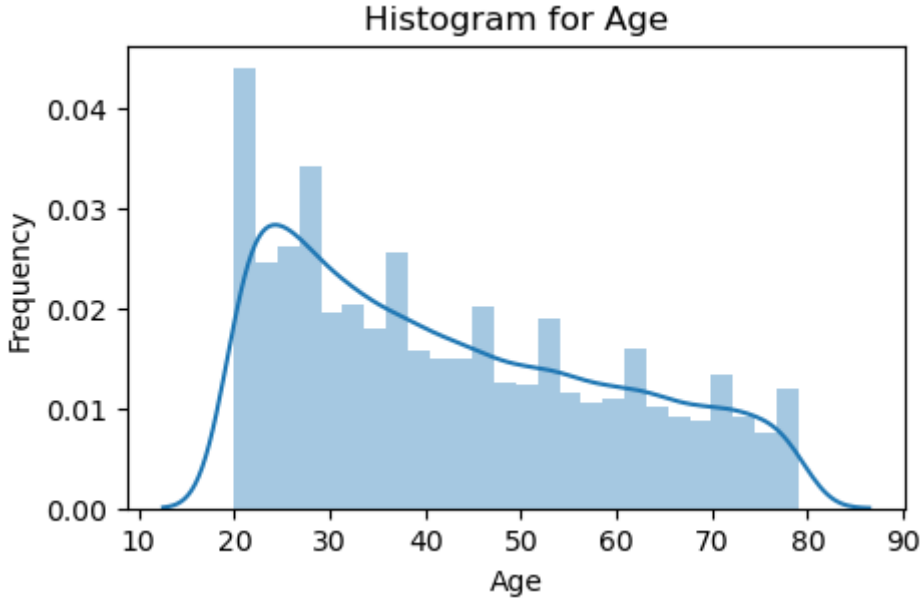
In [12]:
```
check_duplicate(data)
```

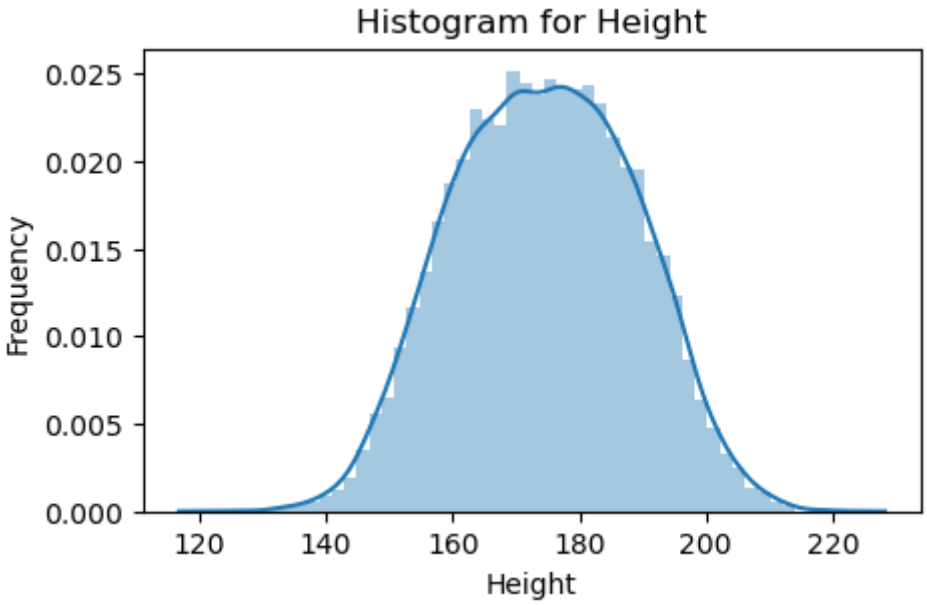
Out[12]:
```
False
```

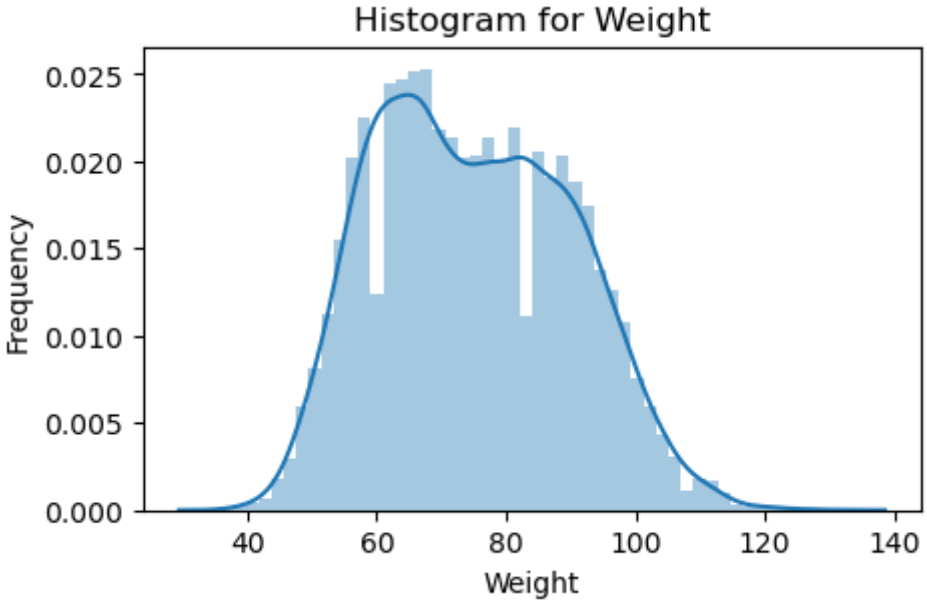
In [15]:
```
plot_graph(data)
```

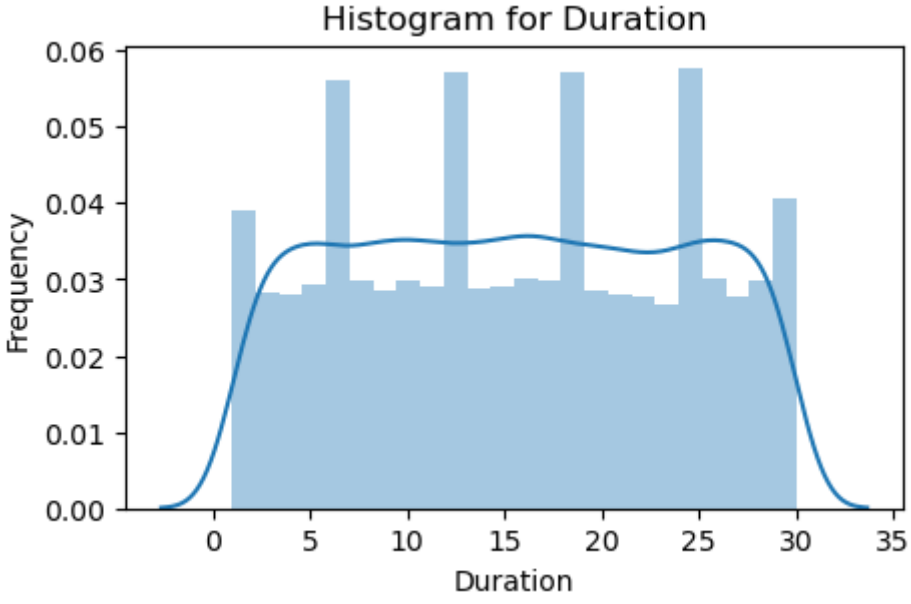

Histogram for User_ID
1e−7
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
1.0
1.2
1.4
1.6
1.8
2.0
User_ID
1e7

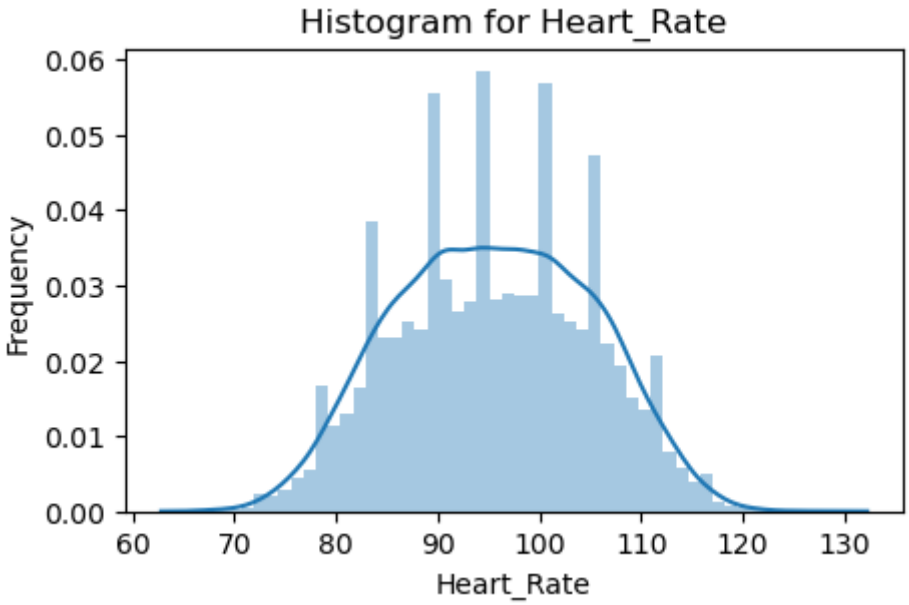

Histogram for Calories
0.008
0.006
0.004
0.002
0.000
Frequency
0
50
100
150
200
250
300
350
Calories

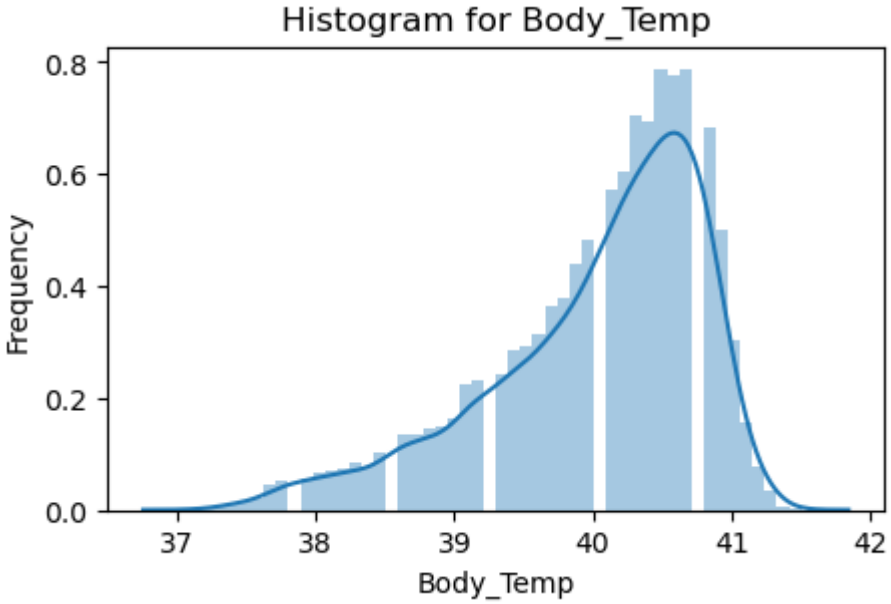

Histogram for Age
0.04
0.03
0.02
0.01
0.00
Frequency
10
20
30
40
50
60
70
80
90
Age


Histogram for Height
Frequency
0.025
0.020
0.015
0.010
0.005
0.000
120
140
160
180
200
220
Height


Histogram for Weight
Frequency
0.025
0.020
0.015
0.010
0.005
0.000
40
60
80
100
120
140
Weight


Histogram for Duration
Frequency
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
5
10
15
20
25
30
35
Duration


Histogram for Heart_Rate
Frequency
0.06
0.05
0.04
0.03
0.02
0.01
0.00
60
70
80
90
100
110
120
130
Heart_Rate


Histogram for Body_Temp
Frequency
0.8
0.6
0.4
0.2
0.0
37
38
39
40
41
42
Body_Temp

In [19]:
```
X,y = seperate_feature_target(data,'Calories')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[19], line 1
----> 1 X,y = seperate_feature_target(data,'Calories')

NameError: name 'seperate_feature_target' is not defined
```

In [20]:
```
x,y = sep_feature_target(data,'Calories')
```

In [21]:
```
x = x.drop(columns=['User_ID'])
```

In [22]:
```
X_train,X_test,y_train,y_test = perform_train_test_split(x, y, test_size=0.20, rand
```

In [23]:
```
preprocessor = ColumnTransformer(transformers=[
    ('ordinal',OrdinalEncoder(),['Gender']),
    ('num',StandardScaler(),['Age',
                             'Height',
                             'Weight',
                             'Duration',
                             'Heart_Rate',
                             'Body_Temp']),
],remainder='passthrough')
```

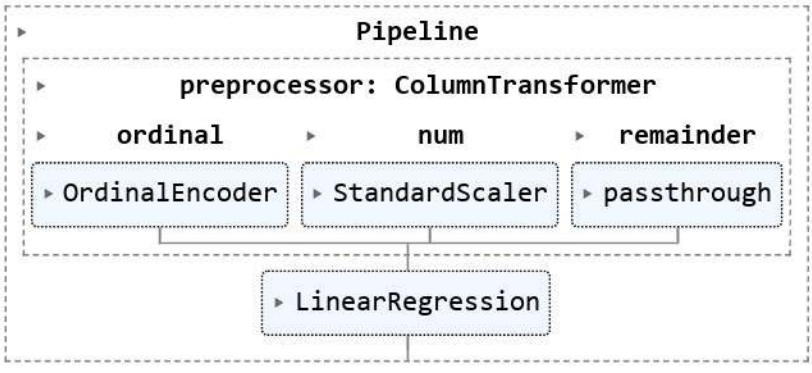
In [24]:
```
pipeline = Pipeline([("preprocessor",preprocessor),
                     ("model",LinearRegression())
                    ])
```

In [25]:
```
from sklearn import set_config
```

In [26]:
```
set_config(display='diagram')
```

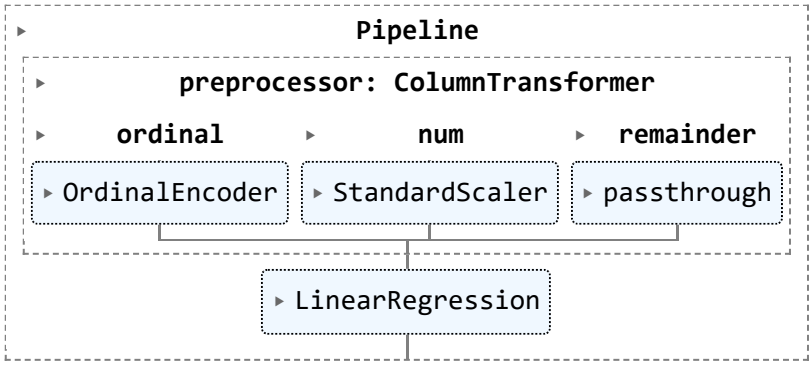
In [27]:
```
pipeline
```

Out[27]:



In [28]:
```
pipeline.fit(X_train,y_train)
```

Out[28]:



In [29]:
```
y_pred = pipeline.predict(X_test)
```

In [30]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[30]:
```
0.9672937151257295
```

In [31]:
```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=5, shuffle=True, random_state=42)
from sklearn.model_selection import cross_val_score
```

In [33]:
```
cv_results = cross_val_score(pipeline, x, y, cv=kfold, scoring='r2')
cv_results.mean()
```

Out[33]:
```
0.9671402283675841
```

In [35]:
```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test,y_pred)
```

Out[35]:
```
8.4415135538497
```

In [42]:
```
def model_scorer(model_name,model):

    output=[]


    output.append(model_name)

    pipeline = Pipeline([
    ('preprocessor',preprocessor),
    ('model',model)])

    X_train,X_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_stat

    pipeline.fit(X_train,y_train)

    y_pred = pipeline.predict(X_test)

    output.append(r2_score(y_test,y_pred))
    output.append(mean_absolute_error(y_test,y_pred))

    kfold = KFold(n_splits=5, shuffle=True, random_state=42)
    cv_results = cross_val_score(pipeline, x, y, cv=kfold, scoring='r2')
    output.append(cv_results.mean())

    return output
```

In [43]:
```
model_dict={
    'log':LinearRegression(),
```

```
    'RF':RandomForestRegressor(),
    'XGBR':XGBRegressor(),
}
```

In [44]:
```
model_output=[]
for model_name,model in model_dict.items():
    model_output.append(model_scorer(model_name,model))
```

In [45]:
```
model_output
```

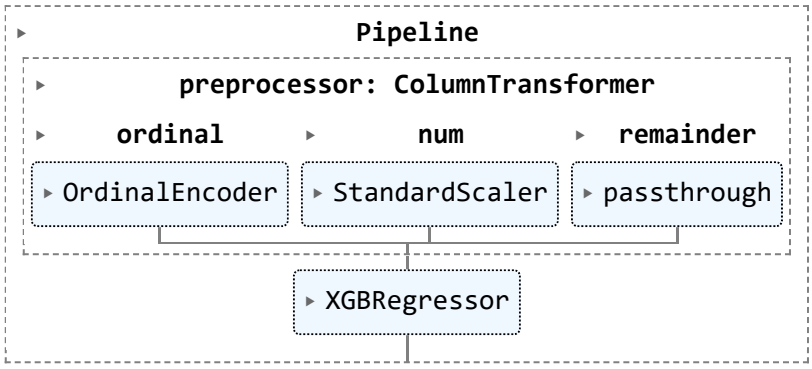
Out[45]:
```
[['log', 0.9672937151257295, 8.4415135538497, 0.9671402283675841],
 ['RF', 0.9982609714089331, 1.6808433333333332, 0.9979005787380586],
 ['XGBR', 0.9988678909361673, 1.4981198125282924, 0.9988510864545181]]
```

In [47]:
```
preprocessor = ColumnTransformer(transformers=[
    ('ordinal',OrdinalEncoder(),['Gender']),
    ('num',StandardScaler(),['Age',
                             'Height',
                             'Weight',
                             'Duration',
                             'Heart_Rate',
                             'Body_Temp']),

],remainder='passthrough')
```

In [48]:
```
pipeline = Pipeline([
    ('preprocessor',preprocessor),
    ('model',XGBRegressor())

])
```

In [50]:
```
pipeline.fit(x,y)
```

Out[50]:



In [51]:
```
sample = pd.DataFrame({
   'Gender':'male',
    'Age':68,
    'Height':190.0,
    'Weight':94.0,
    'Duration':29.0,
    'Heart_Rate':105.0,
    'Body_Temp':40.8,
},index=[0])
```

In [52]:
```
pipeline.predict(sample)
```

Out[52]:
```
array([231.0721], dtype=float32)
```

In [53]:
```
import pickle
with open('pipeline.pkl','wb') as f:
```

```
    pickle.dump(pipeline,f)
with open('pipeline.pkl','rb') as f:
    pipeline_saved = pickle.load(f)
result = pipeline_saved.predict(sample)
```

In [54]:
```
result
```

Out[54]:
```
array([231.0721], dtype=float32)
```

In [ ]:
```
#GUI


import pickle
import pandas as pd
from tkinter import *

def show_entry():

    with open('pipeline.pkl','rb') as f:
        pipeline = pickle.load(f)

    p1 = str(clicked.get())
    p2 = float(e2.get())
    p3 = float(e3.get())
    p4 = float(e4.get())
    p5 = float(e5.get())
    p6 = float(e6.get())
    p7 = float(e7.get())

    sample = pd.DataFrame({
    'Gender':[p1],
    'Age':[p2],
    'Height':[p3],
    'Weight':[p4],
    'Duration':[p5],
    'Heart_Rate':[p6],
    'Body_Temp':[p7],
},index=[0])

    result = pipeline.predict(sample)
    print(result)
    Label(master, text="Amount of Calories Burnt").grid(row=13)
    Label(master, text=result[0]).grid(row=14)


master =Tk()
master.title("Calories Burnt Prediction using Machine Learning")
label = Label(master,text = "Calories Burnt Prediction",bg = "black",
              fg = "white").grid(row=0,columnspan=2)
Label(master,text = "Select Gender").grid(row=1)
Label(master,text = "Enter Your Age").grid(row=2)
Label(master,text = "Enter Your Height").grid(row=3)
Label(master,text = "Enter Your Weight").grid(row=4)
Label(master,text = "Duration").grid(row=5)
Label(master,text = "Heart Rate").grid(row=6)
Label(master,text = "Body Temp").grid(row=7)

clicked = StringVar()
options = ['male', 'female']

e1 = OptionMenu(master , clicked , *options )
e1.configure(width=15)
e2 = Entry(master)
```

```
e3 = Entry(master)
e4 = Entry(master)
e5 = Entry(master)
e6 = Entry(master)
e7 = Entry(master)


e1.grid(row=1,column=1)
e2.grid(row=2,column=1)
e3.grid(row=3,column=1)
e4.grid(row=4,column=1)
e5.grid(row=5,column=1)
e6.grid(row=6,column=1)
e7.grid(row=7,column=1)



Button(master,text="Predict",command=show_entry).grid()

mainloop()
```

In [ ]: