In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Loading the data from csv file
calories = pd.read_csv(r'C:\Users\Johnson\Downloads\Compressed\archive_3\calo
exercise_data = pd.read_csv(r'C:\Users\Johnson\Downloads\Compressed\archive_3
```

In [3]:
```
# print the first 5 rows of the dataframe
calories.head()
```

Out[3]:

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

In [4]:
```
exercise_data.head()
```

Out[4]:

	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

In [5]:
```
#Combining the two Dataframes
calories_data = pd.concat([exercise_data, calories['Calories']], axis=1)
```

In [6]:
```
calories_data.head()
```

Out[6]:

	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

In [7]:
```
calories_data.shape #shape of data
```

Out[7]:
```
(15000, 9)
```

In [8]:
```
calories_data.info() # getting informations about the data
```

```
<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
```

In [9]:
```
calories_data.isnull().sum()  # checking for missing values
```

Out[9]:
```
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

In [10]:
```
calories_data.describe()  #statistical measures about the data
```

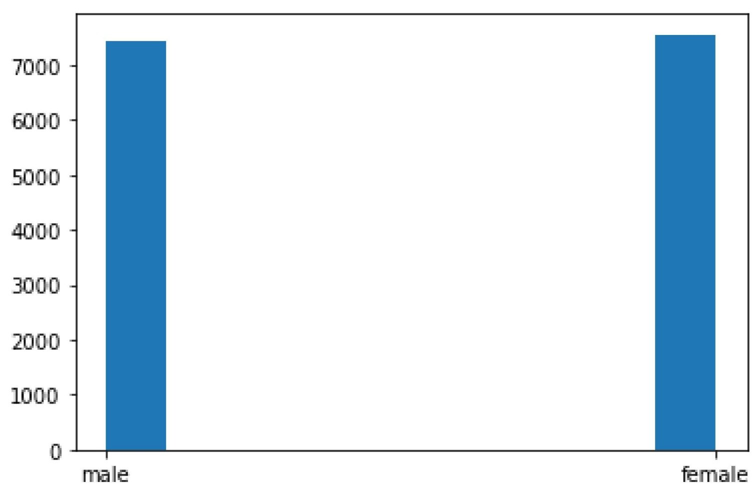
Out[10]:

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

Data Visualization

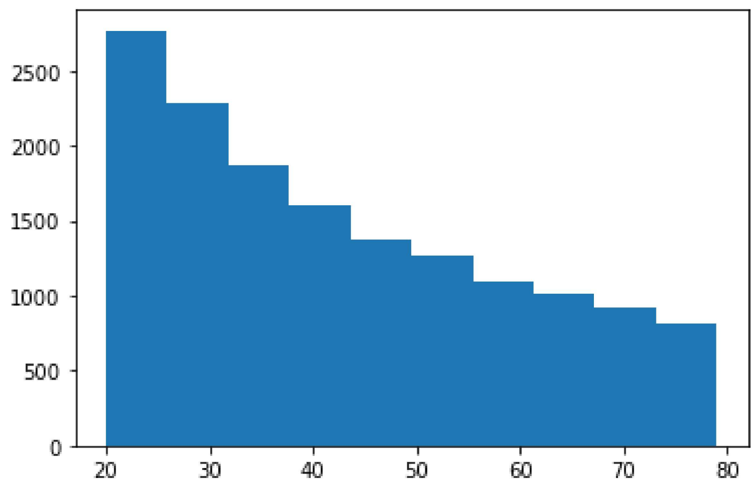
In [11]:
```
# plotting the gender column in count plot
plt.hist(calories_data['Gender'],rwidth=10)
```

Out[11]:
```
(array([7447.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,    0.,
        7553.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



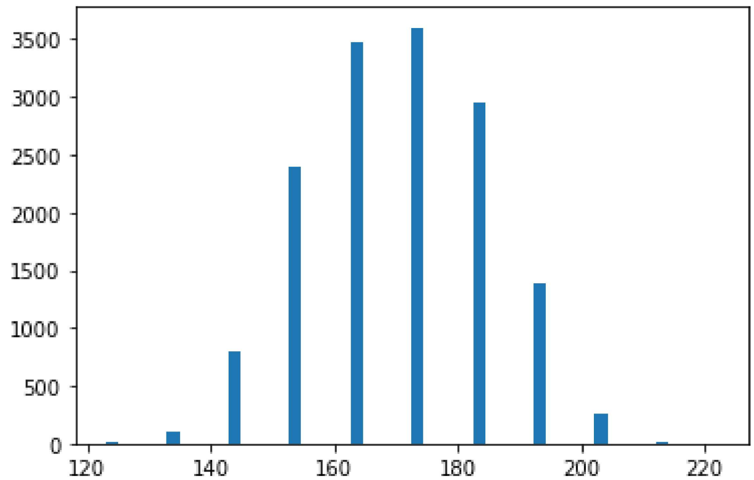
In [12]:
```
# finding the distribution of "Age" column
plt.hist(calories_data['Age'])
```

Out[12]:
```
(array([2770., 2281., 1864., 1606., 1375., 1264., 1100., 1016.,  916.,
         808.]),
 array([20. , 25.9, 31.8, 37.7, 43.6, 49.5, 55.4, 61.3, 67.2, 73.1, 79. ]),
 <BarContainer object of 10 artists>)
```



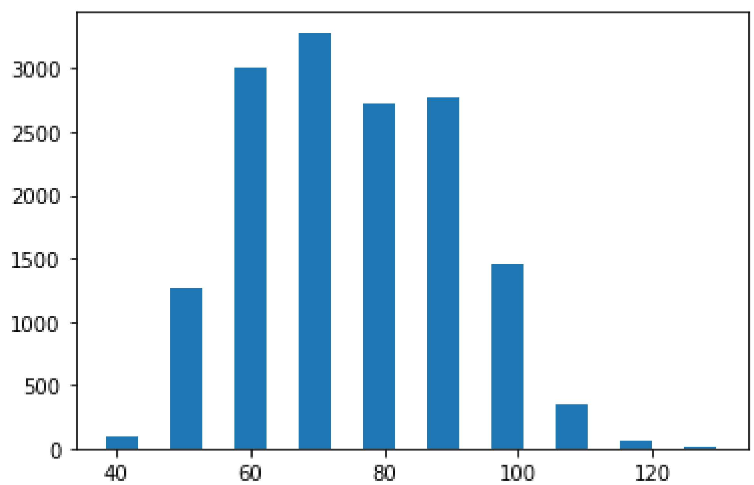
In [13]:
```
# finding the distribution of "Height" column
plt.hist(calories_data['Height'],width=2)
```

Out[13]:
```
(array([   8.,   98.,  800., 2401., 3478., 3600., 2946., 1391.,  262.,
          16.]),
 array([123. , 132.9, 142.8, 152.7, 162.6, 172.5, 182.4, 192.3, 202.2,
        212.1, 222. ]),
 <BarContainer object of 10 artists>)
```



In [14]:
```
# finding the distribution of "Weight" column
plt.hist(calories_data['Weight'],rwidth=0.5)
```

Out[14]:
```
(array([  89., 1261., 3006., 3280., 2729., 2771., 1458.,  343.,   56.,
           7.]),
 array([ 36. ,  45.6,  55.2,  64.8,  74.4,  84. ,  93.6, 103.2, 112.8,
        122.4, 132. ]),
 <BarContainer object of 10 artists>)
```

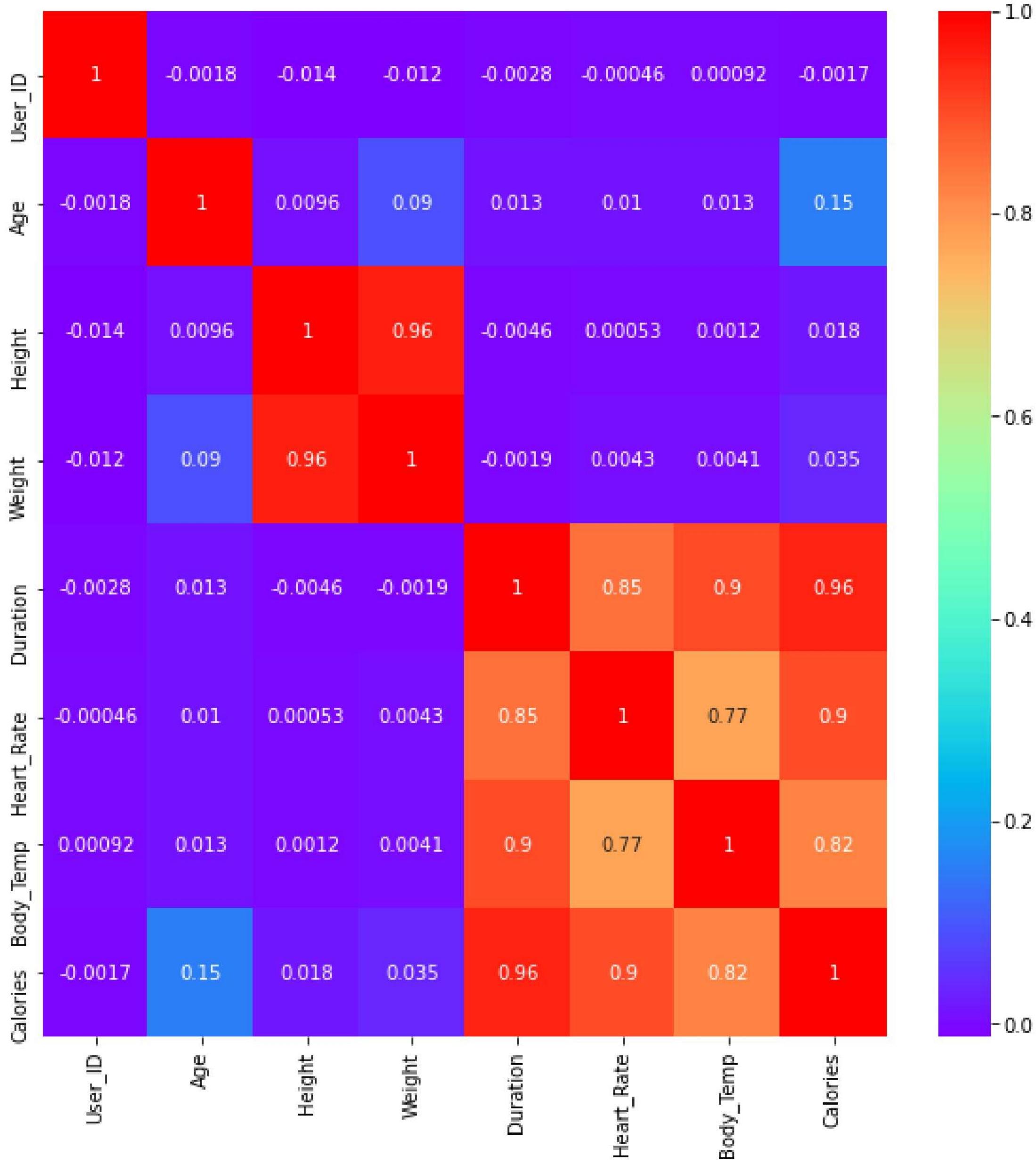


In [15]:
```
correlation = calories_data.corr()
```

In [16]:
```
# constructing a heatmap to understand the correlation

plt.figure(figsize=(10,10))
sns.heatmap(correlation,cbar=True,annot=True,cmap='rainbow')
```

Out[16]: <AxesSubplot:>



In [32]:
```
#Converting the text data to numerical values(Data preprocessing)
calories_data=calories_data.replace({"Gender":{'male':0,'female':1}})
```

In [33]:
```
calories_data.head()
```

Out[33]:

	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

In [34]:
```
#Separating features and Target
X = calories_data.iloc[:,1:8]
y = calories_data.iloc[:,-1]
```

In [35]:
```
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]
```

In [36]:
```
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
```

In [37]:
```
#Splitting the data into training data and Test data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rand
```

In [38]:
```
print(X.shape, X_train.shape, X_test.shape) #to find shape of X,Xtrain,Xtest
```

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

Using SUPPORT VECTOR REGRESSOR

In [39]:
```
# loading the model
from sklearn.svm import SVR
model = SVR()
```

In [40]:
```
# training the model with X_train
model.fit(X_train, y_train)
```

Out[40]:
```
SVR()
```

In [41]:
```
#Prediction on Test Data
test_data_prediction = model.predict(X_test)
```

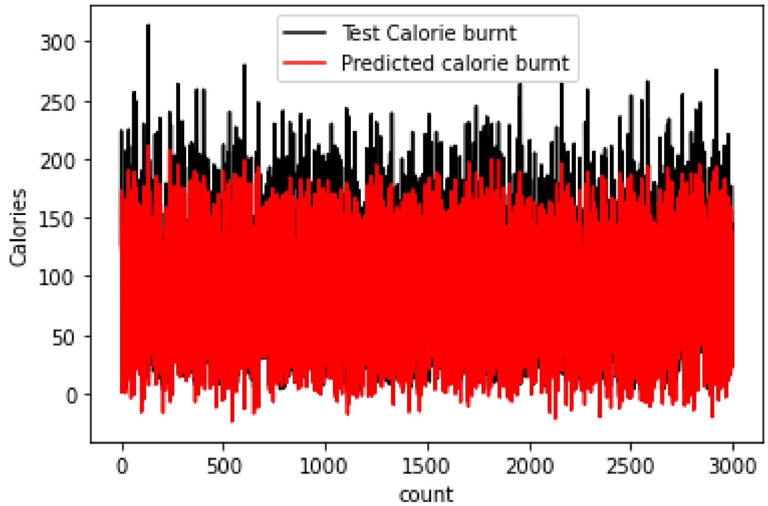
In [42]:
```
print(test_data_prediction)
```

```
[123.04017365 173.117781    56.41704389 ... 138.00300555  22.00311932
  94.25110648]
```

In [43]:
```
#mean square score
from sklearn.metrics import r2_score
score=r2_score(y_test,test_data_prediction)
print(score)
```

```
0.9380560925377789
```

In [44]:
```
#ploting test vs predicted value
y_test=list(y_test)
plt.plot(y_test,color='black',label='Test Calorie burnt')
plt.plot(test_data_prediction,color='red',label='Predicted calorie burnt')
plt.xlabel('count')
plt.ylabel('Calories')
plt.legend()
plt.show()
```



Using Random forest Regressor

In [46]:
```
from sklearn.ensemble import RandomForestRegressor
ran_model=RandomForestRegressor(n_estimators=100)
ran_model.fit(X_train,y_train)
```

Out[46]:
```
RandomForestRegressor()
```

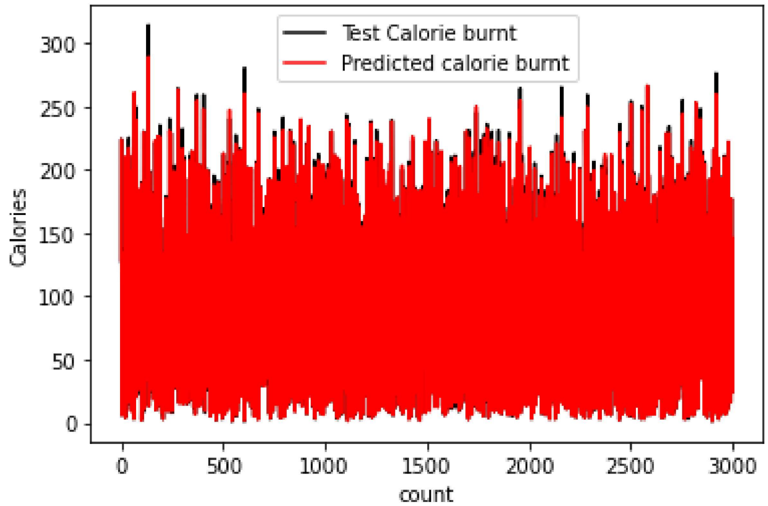
In [47]:
```
random_pred=ran_model.predict(X_test)
```

Score For Random Forest Regressor

In [49]:
```
random_score=r2_score(y_test,random_pred)
print(random_score)
```

```
0.998163190211145
```

In [59]:
```
#ploting test vs predicted value
y_test=list(y_test)
plt.plot(y_test,color='black',label='Test Calorie burnt')
plt.plot(random_pred,color='red',label='Predicted calorie burnt')
plt.xlabel('count')
plt.ylabel('Calories')
plt.legend()
plt.show()
```



Using KNN Regressor

In [50]:
```
from sklearn.neighbors import KNeighborsRegressor
knn_model=KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train,y_train)
```

Out[50]:
```
KNeighborsRegressor()
```

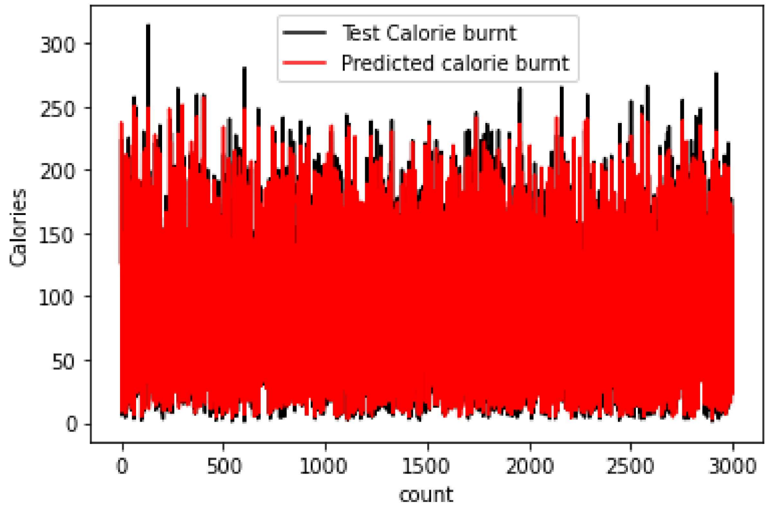
In [51]:
```
knn_pred=knn_model.predict(X_test)
```

Score for KNN Regressor

In [53]:
```
knn_score=r2_score(knn_pred,y_test)
print(knn_score)
```

```
0.9861941161727318
```

In [60]:
```
#ploting test vs predicted value
y_test=list(y_test)
plt.plot(y_test,color='black',label='Test Calorie burnt')
plt.plot(knn_pred,color='red',label='Predicted calorie burnt')
plt.xlabel('count')
plt.ylabel('Calories')
plt.legend()
plt.show()
```



using multiple Linear Regressor

In [54]:
```
from sklearn.linear_model import LinearRegression
lin_model=LinearRegression()
lin_model.fit(X_train,y_train)
```

Out[54]:
```
LinearRegression()
```

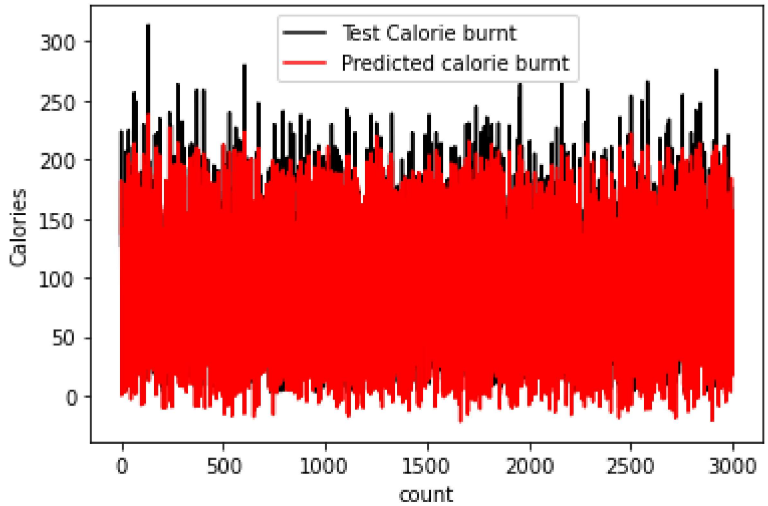
In [56]:
```
lin_pred=lin_model.predict(X_test)
```

Score for Multiple Linear regression

In [58]:
```
lin_score=r2_score(y_test,lin_pred)
print(lin_score)
```

```
0.9668790377181355
```

In [61]:
```
#ploting test vs predicted value
y_test=list(y_test)
plt.plot(y_test,color='black',label='Test Calorie burnt')
plt.plot(lin_pred,color='red',label='Predicted calorie burnt')
plt.xlabel('count')
plt.ylabel('Calories')
plt.legend()
plt.show()
```



Conclusion

Score

Support vector Regressor -93.80

Random Forest Regressor-99.81

K-Nearest Neighbor Regressor-98.61

Multiple Linear Regressor-96.68

It is good to use KNN Regressor for this model

It is good to use KNN Regressor for this model

In [ ]:

```

```