07

April 4, 2025

[1]:

```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

[22]:

```python
# Load the dataset
file_path = "C:\\Users\\HP\\Downloads\\calories.csv"
data = pd.read_csv(file_path)
```

[23]:

```python
# Display basic info
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
```

[24]:

```python
print(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

   Calories
0     231.0
1      66.0
2      26.0
3      71.0
4      35.0
```

[35]:
```
data.describe()
```

[35]:
```
            User_ID        Gender           Age        Height        Weight  \
count  1.500000e+04  15000.000000  15000.000000  15000.000000  15000.000000
mean   1.497736e+07      0.496467     42.789800    174.465133     74.966867
std    2.872851e+06      0.500004     16.980264     14.258114     15.035657
min    1.000116e+07      0.000000     20.000000    123.000000     36.000000
25%    1.247419e+07      0.000000     28.000000    164.000000     63.000000
50%    1.499728e+07      0.000000     39.000000    175.000000     74.000000
75%    1.744928e+07      1.000000     56.000000    185.000000     87.000000
max    1.999965e+07      1.000000     79.000000    222.000000    132.000000

           Duration    Heart_Rate     Body_Temp      Calories
count  15000.000000  15000.000000  15000.000000  15000.000000
mean      15.530600     95.518533     40.025453     89.539533
std        8.319203      9.583328      0.779230     62.456978
min        1.000000     67.000000     37.100000      1.000000
25%        8.000000     88.000000     39.600000     35.000000
50%       16.000000     96.000000     40.200000     79.000000
75%       23.000000    103.000000     40.600000    138.000000
max       30.000000    128.000000     41.500000    314.000000
```

[25]:
```
# Encode categorical 'Gender' column
label_encoder = LabelEncoder()
data['Gender'] = label_encoder.fit_transform(data['Gender'])  # Male=1, Female=0
```

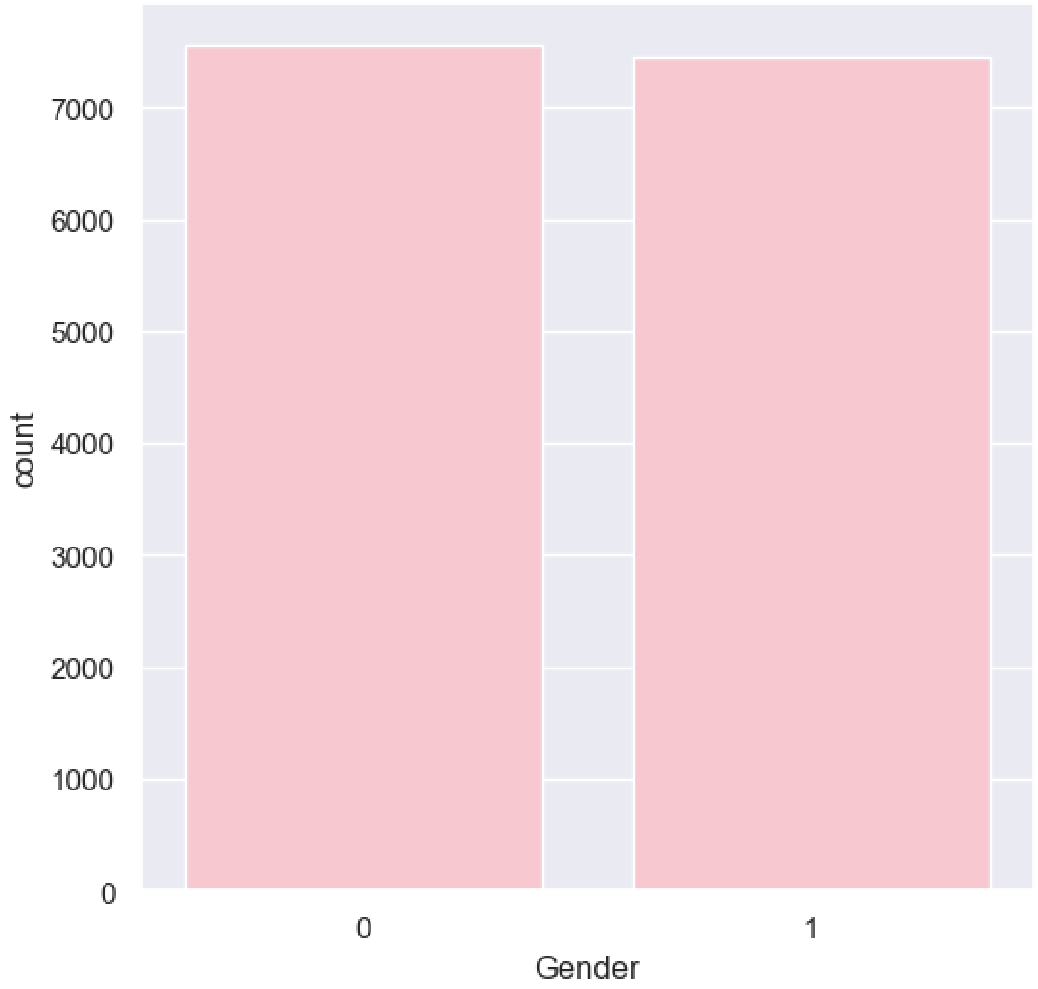
[27]:
```
# Check for missing values
print("Missing Values:\n", data.isnull().sum())
```

```
Missing Values:
 User_ID       0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
```

```
Body_Temp     0
Calories      0
dtype: int64
```

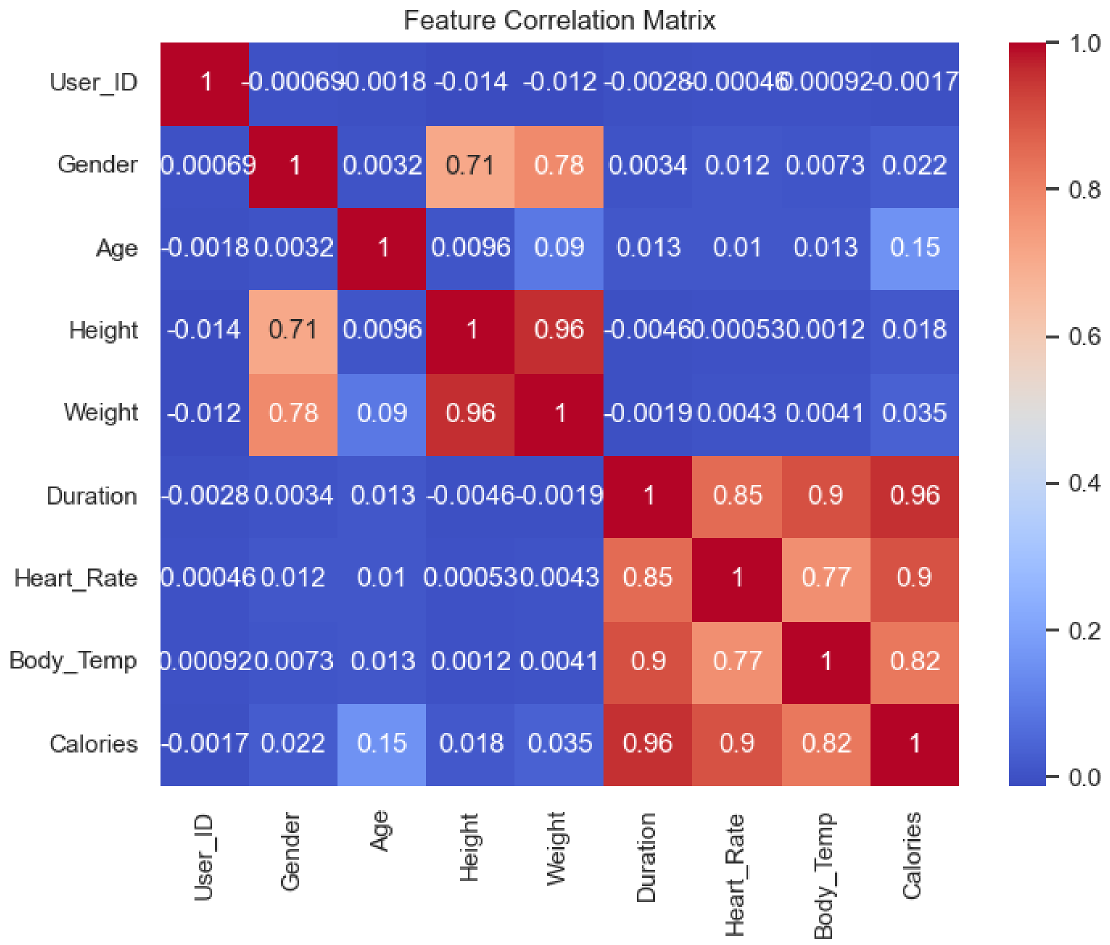
[28]:

```python
sns.set()
plt.figure(figsize=(6,6))
sns.countplot(x=data.Gender,color='pink')
plt.show()
```



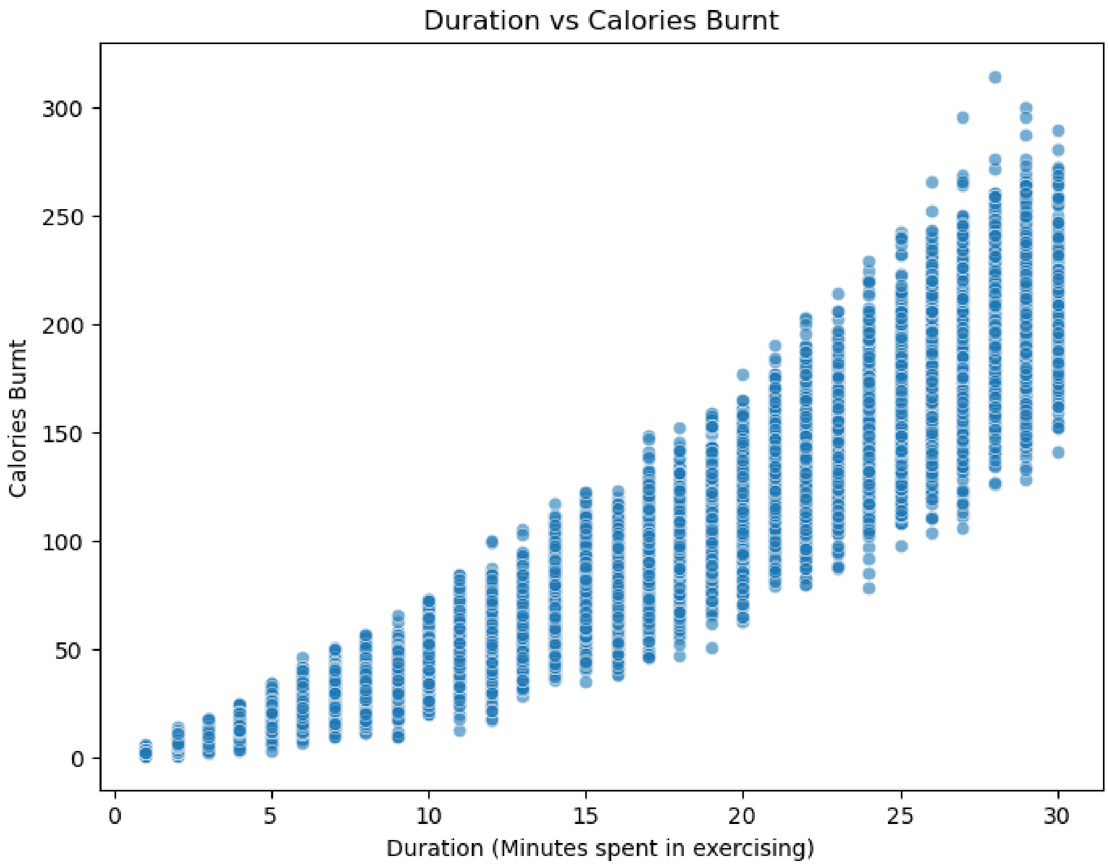
[29]:

```python
# EDA - Visualizing correlations
plt.figure(figsize=(8,6))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Matrix")
plt.show()
```



[7]:

```
# Scatter plot: Duration vs Calories
plt.figure(figsize=(8,6))
sns.scatterplot(x=data['Duration'], y=data['Calories'], alpha=0.6)
plt.xlabel("Duration (Minutes spent in exercising)")
plt.ylabel("Calories Burnt")
plt.title("Duration vs Calories Burnt")
plt.show()
```



[8]:

```
# Define features (X) and target variable (y)
X = data.drop(columns=['User_ID', 'Calories'])  # Exclude ID and target variable
y = data['Calories']
```

[9]:

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[33]:

```
X_train.shape, X_test.shape
```

[33]:

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

[10]:

```
# Train the Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

[10]:

```
LinearRegression()
```

[11]:

```
# Train the Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
rf_model.fit(X_train, y_train)
```

[11]:
```
RandomForestRegressor(random_state=42)
```

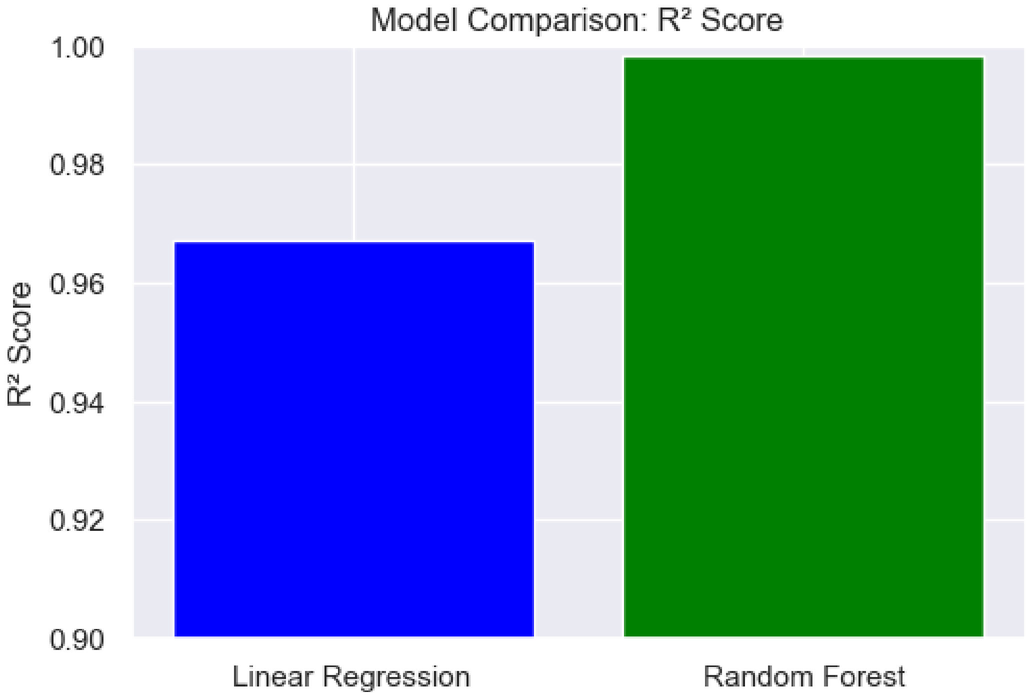
[12]:
```
# Predictions on test data
lr_pred = lr_model.predict(X_test)
rf_pred = rf_model.predict(X_test)
```

[13]:
```
# Model Evaluation
lr_mae = mean_absolute_error(y_test, lr_pred)
rf_mae = mean_absolute_error(y_test, rf_pred)
```

[14]:
```
lr_mse = mean_squared_error(y_test, lr_pred)
rf_mse = mean_squared_error(y_test, rf_pred)

lr_r2 = r2_score(y_test, lr_pred)
rf_r2 = r2_score(y_test, rf_pred)

print(f"Linear Regression Performance:\n MAE: {lr_mae:.2f}\n MSE: {lr_mse:.
 ↪2f}\n R² Score: {lr_r2:.4f}")
print(f"Random Forest Performance:\n MAE: {rf_mae:.2f}\n MSE: {rf_mse:.2f}\n R²␣
 ↪Score: {rf_r2:.4f}")
```

```
Linear Regression Performance:
 MAE: 8.44
 MSE: 132.00
 R² Score: 0.9673
Random Forest Performance:
 MAE: 1.72
 MSE: 7.20
 R² Score: 0.9982
```

[37]:
```
models = ['Linear Regression', 'Random Forest']
r2_scores = [lr_r2, rf_r2]

plt.figure(figsize=(6,4))
plt.bar(models, r2_scores, color=['blue', 'green'])
plt.ylabel('R² Score')
plt.title('Model Comparison: R² Score')
plt.ylim(0.9, 1.0)
plt.show()
```



[36]:

```
if rf_r2 > lr_r2:
    print("\n  Random Forest is the better model!")
else:
    print("\n  Linear Regression is the better model!")
```

```

  Random Forest is the better model!
```

0.1 Random Forest performed better with lower error and a higher R^2 score.

[15]:

```
# Final Test Case - Predict Calories using both models
input_data = (30, 1, 170.0, 70.0, 30.0, 120.0, 36.5)  # Example test case
input_data_as_numpy_array = np.asarray(input_data).reshape(1, -1)

lr_prediction = lr_model.predict(input_data_as_numpy_array)
rf_prediction = rf_model.predict(input_data_as_numpy_array)

print("Predicted Calories Burnt (Linear Regression):", lr_prediction[0])
print("Predicted Calories Burnt (Random Forest):", rf_prediction[0])
```

```
Predicted Calories Burnt (Linear Regression): 231.68019462571664
Predicted Calories Burnt (Random Forest): 223.88
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
```

```
  warnings.warn(
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:493: UserWarning: X does
not have valid feature names, but RandomForestRegressor was fitted with feature
names
  warnings.warn(
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: