In [1]:
```python
import numpy as np
import pandas as pd
```

In [2]:
```python
import os
for dirname, _, filenames in os.walk('CaloriesBurntPrediction'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
CaloriesBurntPrediction/calories.csv
CaloriesBurntPrediction/exercise.csv
```

In [3]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

In [4]:
```python
calories = pd.read_csv('CaloriesBurntPrediction/calories.csv')
exercise = pd.read_csv('CaloriesBurntPrediction/exercise.csv')
```

In [5]:
```python
calories
```

Out[5]:

	User_ID	Calories
0	14733363	231.0
1	14861698	66.0
2	11179863	26.0
3	16180408	71.0
4	17771927	35.0
...	...	...
14995	15644082	45.0
14996	17212577	23.0
14997	17271188	75.0
14998	18643037	11.0
14999	11751526	98.0

15000 rows × 2 columns

In [6]:
```python
exercise
```

Out[6]:

	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
...	...	...	...	...	...	...	...	...
14995	15644082	female	20	193.0	86.0	11.0	92.0	40.4
14996	17212577	female	27	165.0	65.0	6.0	85.0	39.2
14997	17271188	female	43	159.0	58.0	16.0	90.0	40.1
14998	18643037	male	78	193.0	97.0	2.0	84.0	38.3
14999	11751526	male	63	173.0	79.0	18.0	92.0	40.5

15000 rows × 8 columns

In [7]:
```python
data = pd.concat([exercise,calories.Calories],axis=1)
data
```

Out[7]:

	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
...	...	...	...	...	...	...	...	...	...
14995	15644082	female	20	193.0	86.0	11.0	92.0	40.4	45.0
14996	17212577	female	27	165.0	65.0	6.0	85.0	39.2	23.0
14997	17271188	female	43	159.0	58.0	16.0	90.0	40.1	75.0
14998	18643037	male	78	193.0	97.0	2.0	84.0	38.3	11.0
14999	11751526	male	63	173.0	79.0	18.0	92.0	40.5	98.0

15000 rows × 9 columns

In [8]:
```python
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
```

In [9]:
```python
print('The size of Dataframe is: ', data.shape)
print('The Column Name, Record Count and Data Types are as follows: ')
data.info()
```

```
The size of Dataframe is:  (15000, 9)
The Column Name, Record Count and Data Types are as follows:
<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 [10]:
```python
numeric_features = [feature for feature in data.columns if data[feature].dtype != 'O']
categorical_features = [feature for feature in data.columns if data[feature].dtype == 'O']

print('We have {} numerical features : {}'.format(len(numeric_features), numeric_features))
print('\nWe have {} categorical features : {}'.format(len(categorical_features), categorical_features))
```

```
We have 8 numerical features : ['User_ID', 'Age', 'Height', 'Weight', 'Duration', 'Heart_Rate', 'Body_Temp', 'Calorie
s']

We have 1 categorical features : ['Gender']
```

In [11]:
```python
print('Missing Value Presence in different columns of DataFrame are as follows : ')
data.isnull().sum()
```

```
Missing Value Presence in different columns of DataFrame are as follows :
```

Out[11]:
```
User_ID       0
Gender        0
Age           0
Height        0
Weight        0
Duration      0
Heart_Rate    0
Body_Temp     0
Calories      0
dtype: int64
```

In [12]:
```python
print('numerical features')
data.describe()
```

```
numerical features
```

Out[12]:

	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

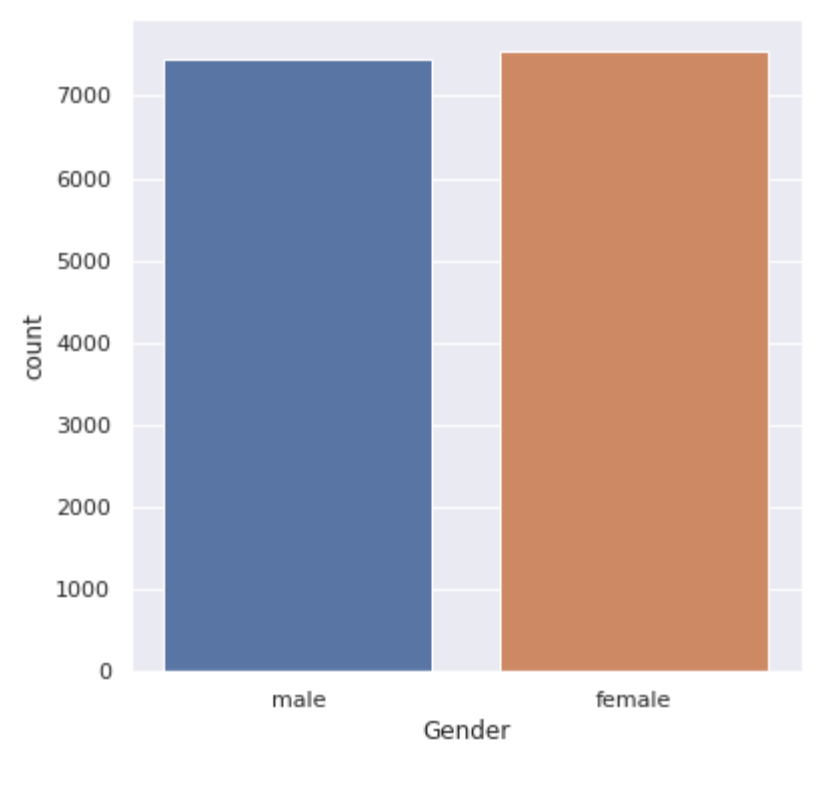
In [13]:
```python
print('categorical features')
data.describe(include='object').T
```

```
categorical features
```

Out[13]:

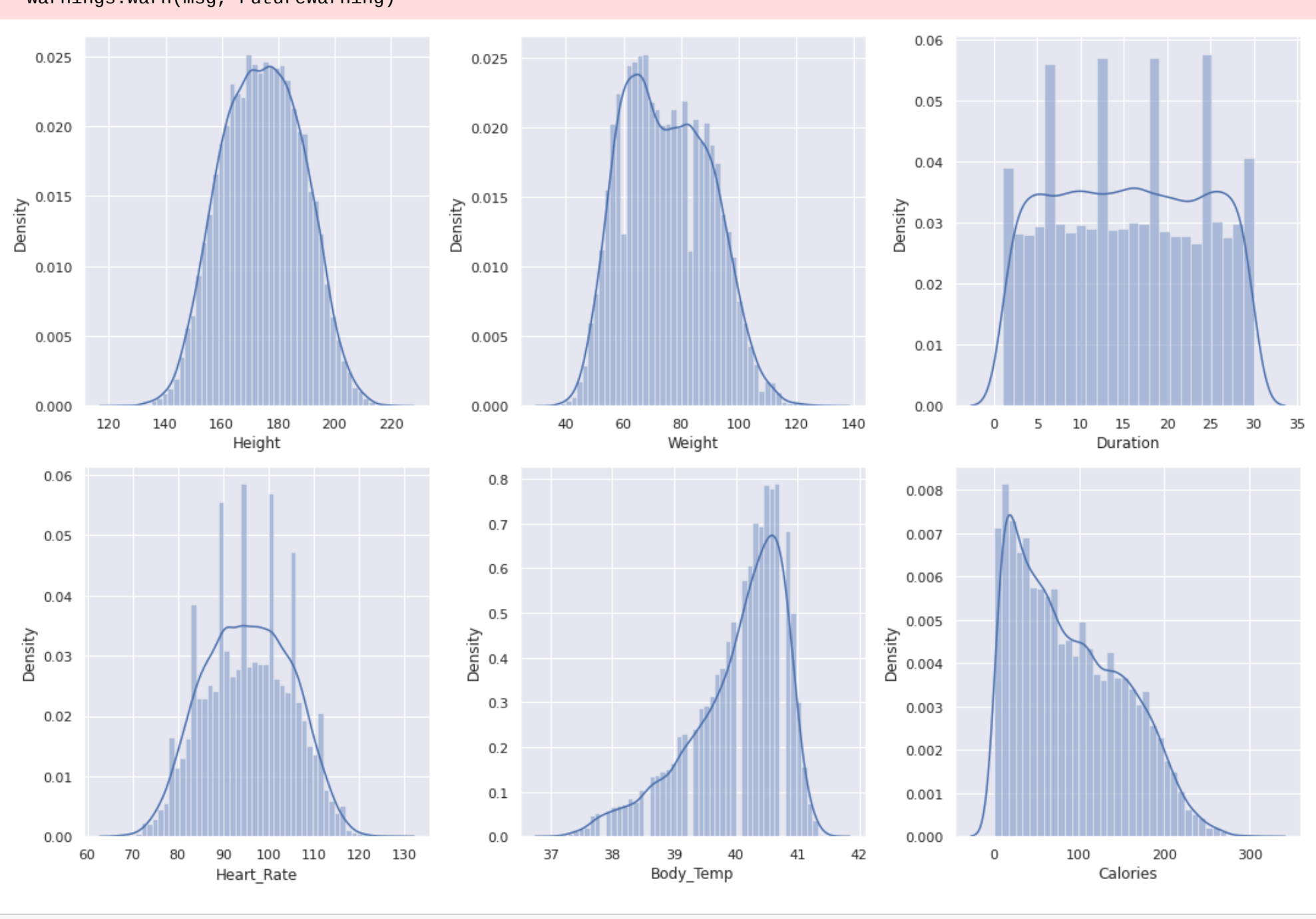
	count	unique	top	freq
Gender	15000	2	female	7553

In [14]:
```python
sns.set()
plt.figure(figsize=(6,6))
sns.countplot(x=data.Gender)
plt.show()
```



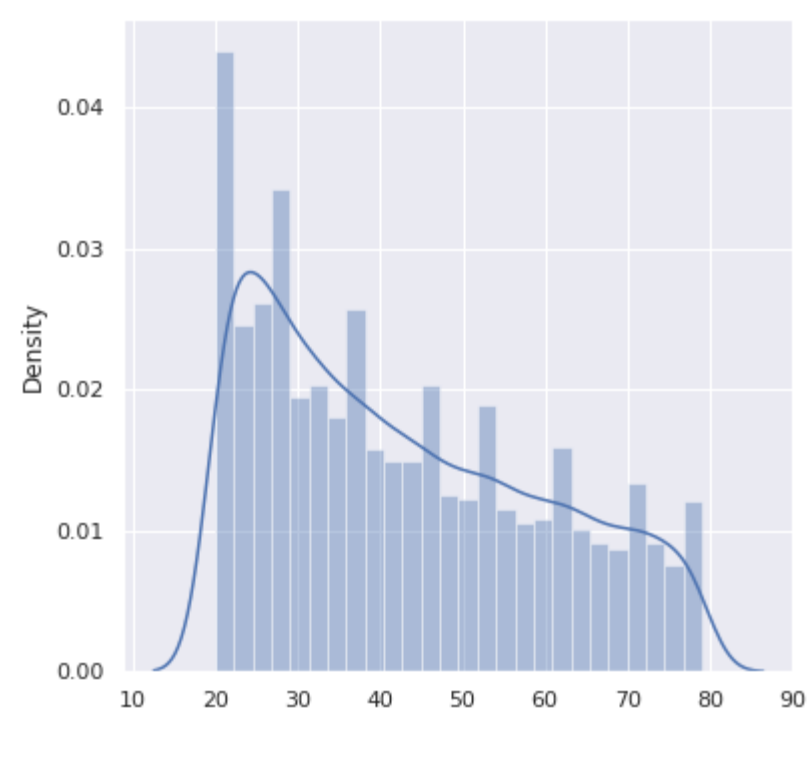
In [15]:
```python
features = data.select_dtypes(include='float').columns

plt.subplots(figsize=(15, 10))
for i, col in enumerate(features):
    plt.subplot(2, 3, i + 1)
    sns.distplot(data[col])
plt.tight_layout()
plt.show()
```

```
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

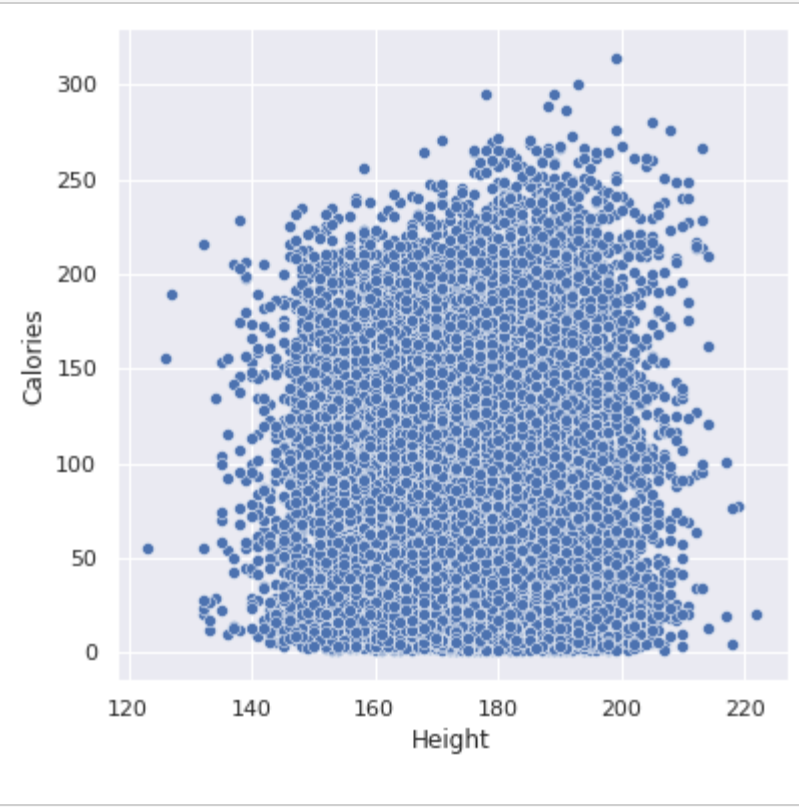


In [16]:
```python
plt.figure(figsize=(6,6))
sns.distplot(x=data.Age)
plt.show()
```

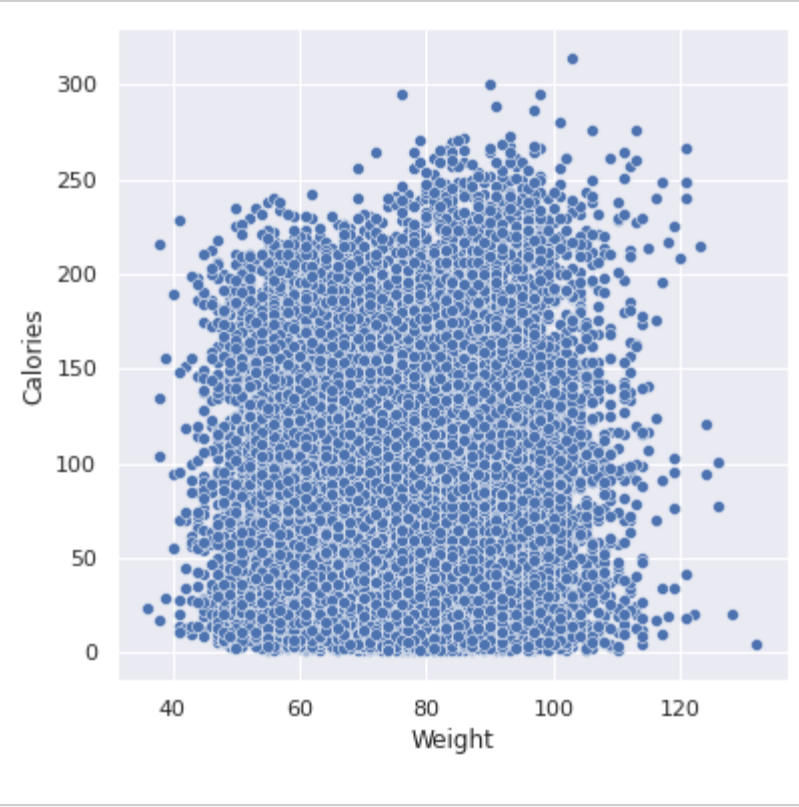
```
/usr/local/anaconda/lib/python3.6/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [17]:
```python
plt.figure(figsize=(6,6))
sns.scatterplot(x=data.Height,y=data.Calories)
plt.show()
```



In [18]:
```python
plt.figure(figsize=(6,6))
sns.scatterplot(x=data.Weight,y=data.Calories)
plt.show()
```



In [19]:
```python
correlation = data.corr()
correlation
```

Out[19]:

	User_ID	Age	Height	Weight	Duration	Heart_Rate	Body_Temp	Calories
User_ID	1.000000	-0.001827	-0.013520	-0.011603	-0.002751	-0.000457	0.000923	-0.001661
Age	-0.001827	1.000000	0.009554	0.090094	0.013247	0.010482	0.013175	0.154395
Height	-0.013520	0.009554	1.000000	0.958451	-0.004625	0.000528	0.001200	0.017537
Weight	-0.011603	0.090094	0.958451	1.000000	-0.001884	0.004311	0.004095	0.035481
Duration	-0.002751	0.013247	-0.004625	-0.001884	1.000000	0.852869	0.903167	0.955421
Heart_Rate	-0.000457	0.010482	0.000528	0.004311	0.852869	1.000000	0.771529	0.897882
Body_Temp	0.000923	0.013175	0.001200	0.004095	0.903167	0.771529	1.000000	0.824558
Calories	-0.001661	0.154395	0.017537	0.035481	0.955421	0.897882	0.824558	1.000000

In [20]:
```python
plt.figure(figsize=(10,6))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f', annot=True, annot_kws={'size':8}, cmap='Blues')
```

Out[20]: <AxesSubplot:>

In [21]:
```python
data.replace({"Gender":{'male':0,'female':1}}, inplace=True)
```

In [22]:
```python
data
```

Out[22]:

	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
...	...	...	...	...	...	...	...	...	...
14995	15644082	1	20	193.0	86.0	11.0	92.0	40.4	45.0
14996	17212577	1	27	165.0	65.0	6.0	85.0	39.2	23.0
14997	17271188	1	43	159.0	58.0	16.0	90.0	40.1	75.0
14998	18643037	0	78	193.0	97.0	2.0	84.0	38.3	11.0
14999	11751526	0	63	173.0	79.0	18.0	92.0	40.5	98.0

15000 rows × 9 columns

In [23]:
```python
X = data.drop(columns = ['User_ID','Calories'], axis=1)
y = data['Calories']
```

In [24]:
```python
X
```

Out[24]:

	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 × 7 columns

In [25]:
```python
y
```

Out[25]:
```
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 [26]:
```python
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
```

In [27]:
```python
scaler.fit(X)
```

Out[27]: StandardScaler(copy=True, with_mean=True, with_std=True)

In [28]:
```python
standardized_data = scaler.transform(X)
```

In [29]:
```python
standardized_data
```

Out[29]:
```
array([[-1.00709181,  1.48472604,  1.08958204, ...,  1.61912727,
         0.98940395,  0.99402302],
       [ 0.99295813, -1.34217934, -0.59372619, ..., -0.1839901 ,
        -0.15846102,  0.3523425 ],
       [-1.00709181,  1.5436199 ,  0.31806577, ..., -1.26586052,
        -0.78456918, -1.70103518],
       ...,
       [ 0.99295813,  0.01237949, -1.08469109, ...,  0.05642555,
        -0.57586646,  0.09567029],
       [-1.00709181,  2.07366466,  1.29999557, ..., -1.62648399,
        -1.20197462, -2.21437961],
       [-1.00709181,  1.19025673, -0.10276129, ...,  0.2968412 ,
        -0.36716374,  0.60901471]])
```

In [30]:
```python
X = standardized_data
X
```

Out[30]:
```
array([[-1.00709181,  1.48472604,  1.08958204, ...,  1.61912727,
         0.98940395,  0.99402302],
       [ 0.99295813, -1.34217934, -0.59372619, ..., -0.1839901 ,
        -0.15846102,  0.3523425 ],
       [-1.00709181,  1.5436199 ,  0.31806577, ..., -1.26586052,
        -0.78456918, -1.70103518],
       ...,
       [ 0.99295813,  0.01237949, -1.08469109, ...,  0.05642555,
        -0.57586646,  0.09567029],
       [-1.00709181,  2.07366466,  1.29999557, ..., -1.62648399,
        -1.20197462, -2.21437961],
       [-1.00709181,  1.19025673, -0.10276129, ...,  0.2968412 ,
        -0.36716374,  0.60901471]])
```

In [31]:
```python
from sklearn.model_selection import train_test_split, RandomizedSearchCV

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [32]:
```python
X.shape,X_train.shape,X_test.shape
```

Out[32]: ((15000, 7), (12000, 7), (3000, 7))

In [33]:
```python
y.shape, y_train.shape, y_test.shape
```

Out[33]: ((15000,), (12000,), (3000,))

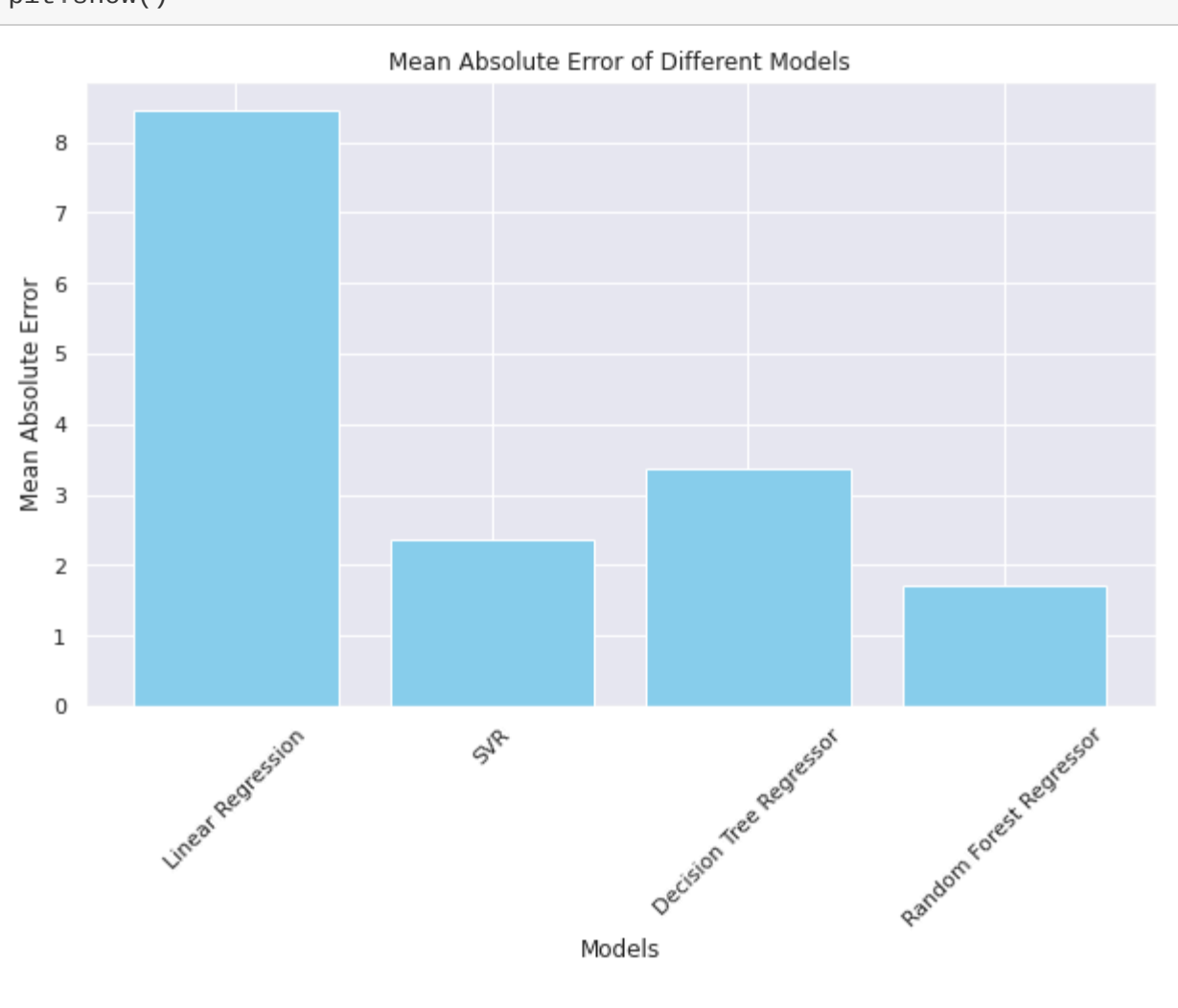
In [34]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

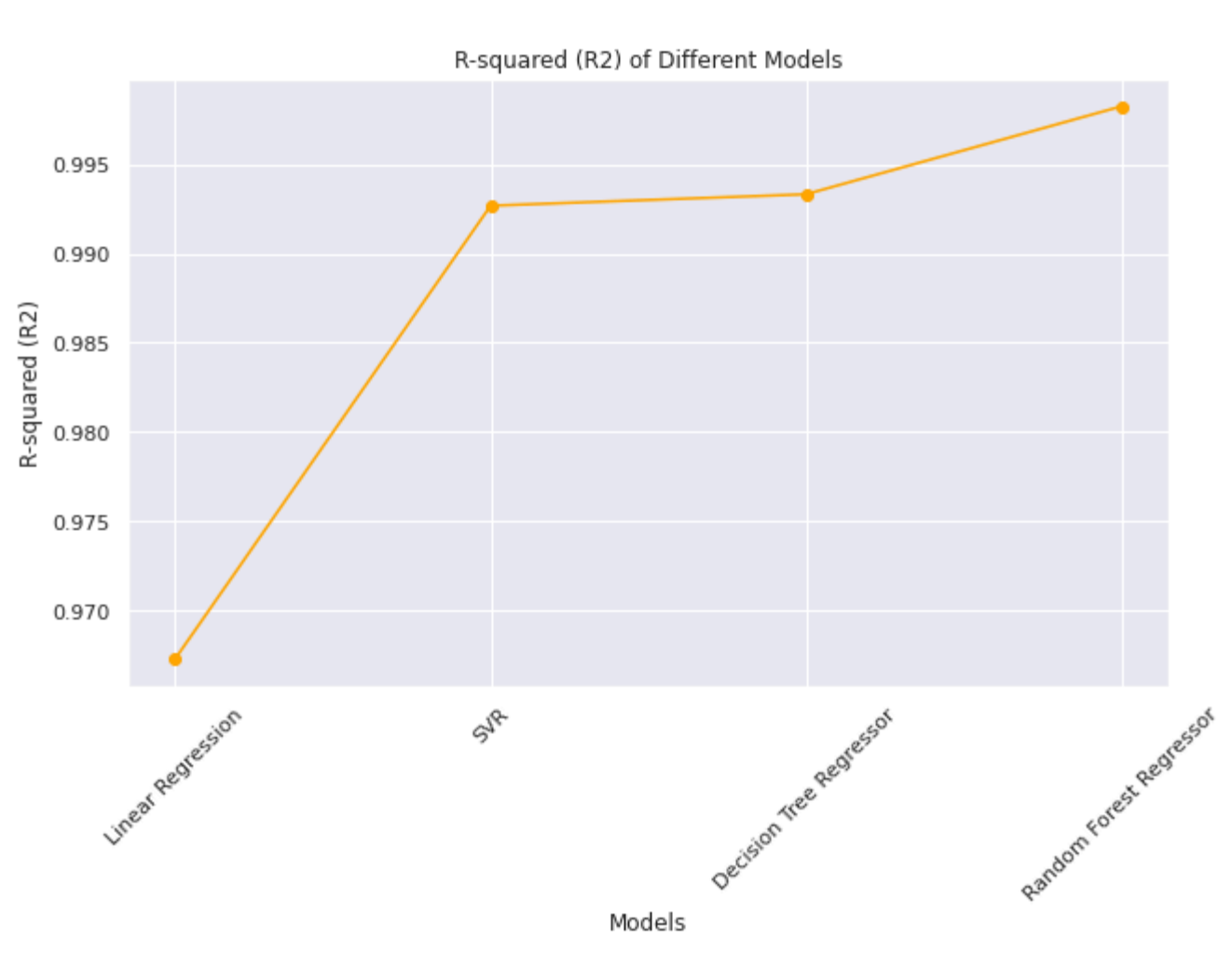
In [35]:
```python
models = [LinearRegression, SVR, DecisionTreeRegressor, RandomForestRegressor]
mae_scores = []
mse_scores = []
rmse_scores = []
r2_scores = []

for model in models:
    regressor = model().fit(X_train, y_train)
    y_pred = regressor.predict(X_test)

    mae_scores.append(mean_absolute_error(y_test, y_pred))
    mse_scores.append(mean_squared_error(y_test, y_pred))
    rmse_scores.append(mean_squared_error(y_test, y_pred, squared=False))
    r2_scores.append(r2_score(y_test, y_pred))

regression_metrics_df = pd.DataFrame({
    "Model": ["Linear Regression", "SVR", "Decision Tree Regressor", "Random Forest Regressor"],
    "Mean Absolute Error": mae_scores,
    "Mean Squared Error": mse_scores,
    "Root Mean Squared Error": rmse_scores,
    "R-squared (R2)": r2_scores
})

regression_metrics_df.set_index('Model', inplace=True)
regression_metrics_df
```

Out[35]:

Model	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	R-squared (R2)
Linear Regression	8.441514	131.995746	11.488940	0.967294
SVR	2.365911	29.571061	5.437928	0.992673
Decision Tree Regressor	3.466667	28.652000	5.352756	0.992901
Random Forest Regressor	1.691927	6.997676	2.645312	0.998266

In [36]:
```python
# Model Comparison & Evaluation (Bar Chart)
models = ['Linear Regression','SVR', 'Decision Tree Regressor', 'Random Forest Regressor']
mae_scores = [8.441514, 2.365911,3.372333,1.700753]
r2_scores = [0.967294, 0.992673, 0.993313,0.998239]
plt.figure(figsize=(10, 6))
plt.bar(models, mae_scores, color='skyblue')
plt.xlabel('Models')
plt.ylabel('Mean Absolute Error')
plt.title('Mean Absolute Error of Different Models')
plt.xticks(rotation=45)
plt.show()

# Model Comparison & Evaluation (Line Chart)
plt.figure(figsize=(10, 6))
plt.plot(models, r2_scores, marker='o', color='orange')
plt.xlabel('Models')
plt.ylabel('R-squared (R2)')
plt.title('R-squared (R2) of Different Models')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```





In [37]:
```python
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Out[37]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [38]:
```python
pred = lr.predict(X_train)
score = metrics.r2_score(y_train,pred)
score
```

Out[38]: 0.9671621074066676

In [39]:
```python
pred1 = lr.predict(X_test)
score1 = metrics.r2_score(y_test,pred1)
score1
```

Out[39]: 0.9672937151257295

In [40]:
```python
dt = DecisionTreeRegressor()
dt.fit(X_train,y_train)
pred = dt.predict(X_train)
score = metrics.r2_score(y_train,pred)
score
```

Out[40]: 1.0

In [41]:
```python
pred1 = dt.predict(X_test)
score1 = metrics.r2_score(y_test,pred1)
score1
```

Out[41]: 0.9932772377316694

In [42]:
```python
rf = RandomForestRegressor()
rf.fit(X_train,y_train)
pred = rf.predict(X_train)
score = metrics.r2_score(y_train,pred)
score
```

Out[42]: 0.999690315732729

In [43]:
```python
pred1 = rf.predict(X_test)
score1 = metrics.r2_score(y_test,pred1)
score1
```

Out[43]: 0.9982105884209925

In [45]:
```python
pred1 = rf.predict(X_test)
pred1
```

Out[45]: array([170.81, 192.12, 53.04, ..., 120.11, 19.81, 213.08])

In [46]:
```python
pred1 = lr.predict(X_test)
pred1
```

Out[46]:
```
array([170.67646027, 192.08857813,  56.14796403, ..., 129.66570069,
         2.4152784 , 191.99081681])
```

In [47]:
```python
pred1 = dt.predict(X_test)
pred1
```

Out[47]: array([166., 188., 52., ..., 123., 20., 207.])

In [49]:
```python
sv=SVR()
sv.fit(X_train,y_train)
pred = sv.predict(X_train)
pred1 = sv.predict(X_test)
pred1
```

Out[49]:
```
array([172.81552926, 179.15558795,  53.31299312, ..., 119.72146864,
        20.95246579, 198.13573399])
```

In [ ]: