In [23]:

```python
import pandas as pd
```

In [24]:

```python
df = pd.read_csv('cardio_sample_dataset.csv')
```

In [25]:

```python
df.head()
```

Out[25]:

	age	gender	height (cm)	weight (kg)	SBP	DBP	cholesterol	glucose	smoking	alcohol	physical_active	cardio
0	48	1	156	56.0	100	60	1	1	0	0	0	0
1	60	1	151	67.0	120	80	2	2	0	0	0	0
2	61	1	157	93.0	130	80	3	1	0	0	1	0
3	48	1	158	71.0	110	70	1	1	0	0	1	0
4	54	1	164	68.0	110	60	1	1	0	0	0	0

In [26]:

```python
df.shape
```

Out[26]:

```
(3000, 12)
```

In [27]:

```python
df.isnull().sum()
```

Out[27]:

```
age                0
gender             0
height (cm)        0
weight (kg)        0
SBP                0
DBP                0
cholesterol        0
glucose            0
smoking            0
alcohol            0
physical_active    0
cardio             0
dtype: int64
```

In [28]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   age              3000 non-null   int64  
 1   gender           3000 non-null   int64  
 2   height (cm)      3000 non-null   int64  
 3   weight (kg)      3000 non-null   float64
 4   SBP              3000 non-null   int64  
 5   DBP              3000 non-null   int64  
 6   cholesterol      3000 non-null   int64  
 7   glucose          3000 non-null   int64  
 8   smoking          3000 non-null   int64  
 9   alcohol          3000 non-null   int64  
 10  physical_active  3000 non-null   int64  
 11  cardio           3000 non-null   int64  
```

```
dtypes: float64(1), int64(11)
memory usage: 281.4 KB
```

In [50]:

```
df.describe()
```

Out[50]:

	age	gender	height (cm)	weight (kg)	SBP	DBP	cholesterol	glucose	
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.00000	3000.000000	3000.000000	3000.000000	300
mean	53.231000	1.500000	165.714333	75.171600	127.09200	97.481667	1.363333	1.226667	
std	6.899578	0.500083	8.122934	14.890697	26.97986	211.475497	0.682748	0.571604	
min	39.000000	1.000000	76.000000	40.000000	11.00000	0.000000	1.000000	1.000000	
25%	48.000000	1.000000	160.000000	65.000000	120.00000	80.000000	1.000000	1.000000	
50%	54.000000	1.500000	166.000000	73.000000	120.00000	80.000000	1.000000	1.000000	
75%	58.000000	2.000000	170.000000	84.000000	140.00000	90.000000	1.000000	1.000000	
max	65.000000	2.000000	198.000000	200.000000	906.00000	10000.000000	3.000000	3.000000	

In [29]:

```
X = df.drop(['cardio'],axis='columns')
y = df.cardio
```

In [30]:

```
X
```

Out[30]:

	age	gender	height (cm)	weight (kg)	SBP	DBP	cholesterol	glucose	smoking	alcohol	physical_active
0	48	1	156	56.0	100	60	1	1	0	0	0
1	60	1	151	67.0	120	80	2	2	0	0	0
2	61	1	157	93.0	130	80	3	1	0	0	1
3	48	1	158	71.0	110	70	1	1	0	0	1
4	54	1	164	68.0	110	60	1	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
2995	40	2	171	111.0	130	90	1	1	0	0	0
2996	52	2	172	88.0	160	90	1	1	0	0	1
2997	62	2	175	73.0	146	89	2	2	0	0	1
2998	52	2	175	94.0	170	110	3	3	1	0	0
2999	54	2	175	97.0	160	100	2	1	0	0	1

3000 rows × 11 columns

In [31]:

```
y
```

Out[31]:

```
0       0
1       0
2       0
3       0
4       0
       ..
2995    1
```

```
2996    1
2997    1
2998    1
2999    1
Name: cardio, Length: 3000, dtype: int64
```

In [34]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[34]:

```
array([[-0.75828873, -1.        , -1.19611373, ..., -0.36635434,
        -0.2694026 , -1.99584286],
       [ 0.98123808, -1.        , -1.81175749, ..., -0.36635434,
        -0.2694026 , -1.99584286],
       [ 1.12619864, -1.        , -1.07298498, ..., -0.36635434,
        -0.2694026 ,  0.50104145],
       ...,
       [ 1.27115921,  1.        ,  1.14333254, ..., -0.36635434,
        -0.2694026 ,  0.50104145],
       [-0.17844646,  1.        ,  1.14333254, ...,  2.72959781,
        -0.2694026 , -1.99584286],
       [ 0.11147468,  1.        ,  1.14333254, ..., -0.36635434,
        -0.2694026 ,  0.50104145]])
```

In [35]:

```
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, random_state=42)
```

In [37]:

```
X_train.shape
```

Out[37]:

```
(2400, 11)
```

In [38]:

```
X_test.shape
```

Out[38]:

```
(600, 11)
```

In [49]:

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DTC = DT.fit(X_train,y_train)
DTC.score(X_test,y_test)
```

Out[49]:

```
0.6333333333333333
```

In [39]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(DecisionTreeClassifier(),X,y,cv=5)
print(scores)
scores.mean()
```

```
[0.67666667 0.64666667 0.64       0.63166667 0.64      ]
```

Out[39]:

```
0.647
```

In [42]:

```
from sklearn import svm
S = svm.SVC()
support = S.fit(X_train,y_train)
support.score(X_test,y_test)
```

```
Out[42]: 0.7083333333333334
```

```
In [43]:
scores = cross_val_score(svm.SVC(),X,y,cv=5)
print(scores)
scores.mean()
```

```
[0.72166667 0.705      0.69666667 0.69666667 0.715     ]
Out[43]: 0.7070000000000001
```

```
In [44]:
from sklearn.naive_bayes import GaussianNB
GNB = GaussianNB()
nb = GNB.fit(X_train,y_train)
nb.score(X_test,y_test)
```

```
Out[44]: 0.6016666666666667
```

```
In [45]:
scores = cross_val_score(GaussianNB(),X,y,cv=5)
print(scores)
scores.mean()
```

```
[0.615      0.585      0.57       0.60333333 0.565     ]
Out[45]: 0.5876666666666667
```