In [ ]:
```
### unsupervised learning
### Dimensionality reduction
### Principal component analysis (PCA)
```

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:
```
import pandas as pd
```

In [5]:
```
from sklearn.datasets import load_breast_cancer
```

In [6]:
```
cancer = load_breast_cancer()
```

In [7]:
```
cancer.keys()
```

Out[7]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filenam
e', 'data_module'])
```

In [8]:
```
df= pd.DataFrame(cancer['data'] ,columns = cancer['feature_names'])
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

Out[10]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...

5 rows × 30 columns

In [13]:

```
### check null value

df.isna().sum()
```

Out[13]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
dtype: int64
```

In [11]:

```
### Make data into StanderedScal
from sklearn.preprocessing import StandardScaler
```

In [12]:

```
new_data = StandardScaler()
new_data.fit(df)
```

Out[12]:

```
▼ StandardScaler
StandardScaler()
```

In [13]:

```
scaled_data = new_data.transform(df)
```

In [14]:
```
scaled_data
```

Out[14]:
```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

In [15]:
```
### apply PCA model
from sklearn.decomposition import PCA
```

In [16]:
```
pca = PCA(n_components=5)
```

In [17]:
```
pca.fit(scaled_data)
```

Out[17]:
```
▼        PCA
PCA(n_components=5)
```

In [18]:
```
x_pca = pca.transform(scaled_data)
```

In [19]:
```
x_pca
```

Out[19]:
```
array([[ 9.19283683,  1.94858307, -1.12316615,  3.63373081, -1.19511005],
       [ 2.3878018 , -3.76817174, -0.5292927 ,  1.11826393,  0.62177491],
       [ 5.73389628, -1.0751738 , -0.55174757,  0.91208256, -0.17708579],
       ...,
       [ 1.25617928, -1.90229671,  0.56273052, -2.08922698,  1.80999129],
       [10.37479406,  1.67201011, -1.87702933, -2.35603112, -0.03374194],
       [-5.4752433 , -0.67063679,  1.49044313, -2.29915733, -0.18470313]])
```

In [20]:
```
### shape befor apply PCA
scaled_data.shape
```

Out[20]:
```
(569, 30)
```

In [21]:
```
### shape after apply PCA
x_pca.shape
```

Out[21]:
```
(569, 5)
```

In [22]:
```
df1 = pd.DataFrame(x_pca)
```

In [23]:
```
df1
```

Out[23]:

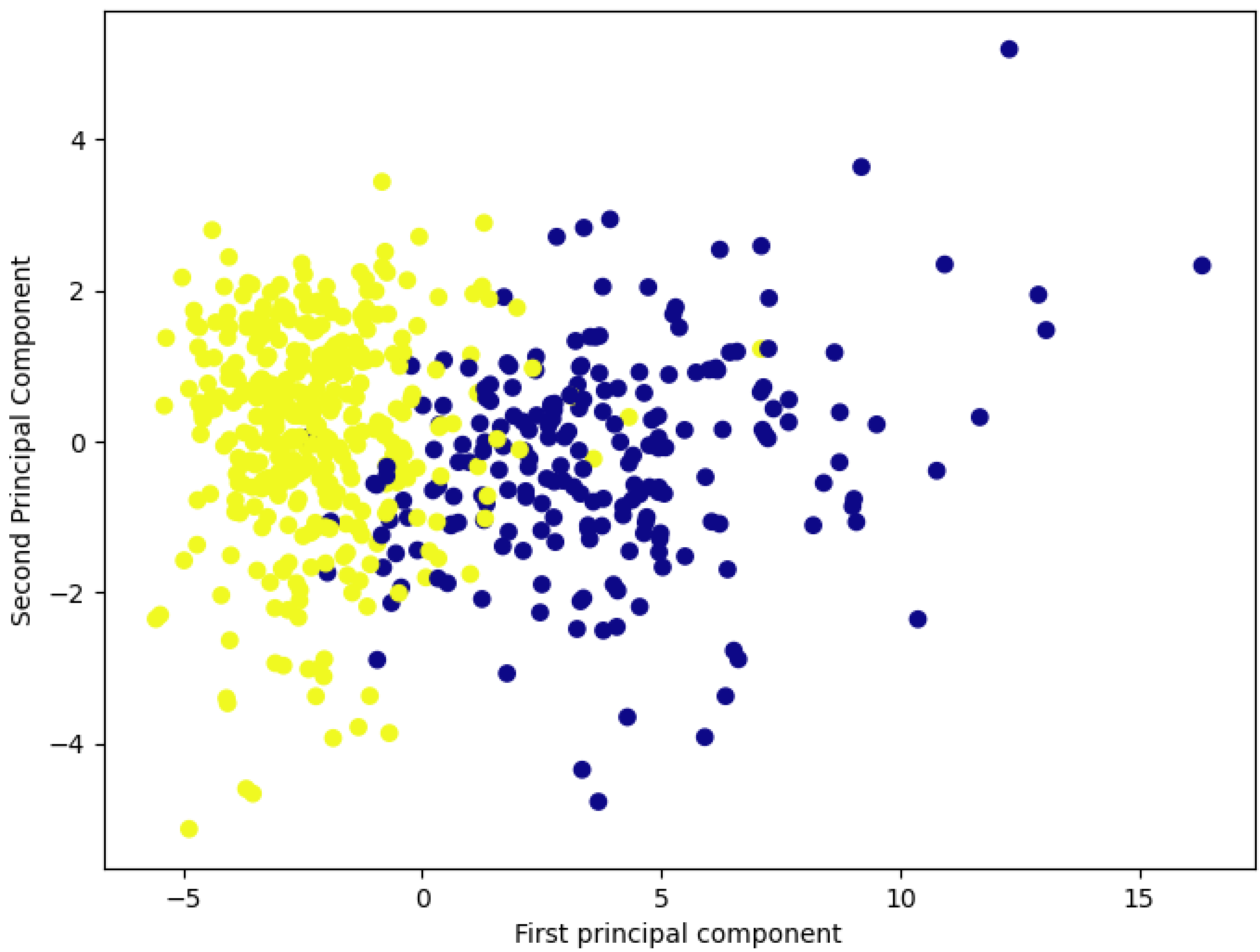
	0	1	2	3	4
0	9.192837	1.948583	-1.123166	3.633731	-1.195110
1	2.387802	-3.768172	-0.529293	1.118264	0.621775
2	5.733896	-1.075174	-0.551748	0.912083	-0.177086
3	7.122953	10.275589	-3.232790	0.152547	-2.960878
4	3.935302	-1.948072	1.389767	2.940639	0.546747
...	...	...	...	...	...
564	6.439315	-3.576817	2.459487	1.177314	-0.074824
565	3.793382	-3.584048	2.088476	-2.506028	-0.510723
566	1.256179	-1.902297	0.562731	-2.089227	1.809991
567	10.374794	1.672010	-1.877029	-2.356031	-0.033742
568	-5.475243	-0.670637	1.490443	-2.299157	-0.184703

569 rows × 5 columns

In [26]:

```
### scatter plot
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0] ,x_pca[: ,3] ,c = cancer['target'] ,cmap ='plasma')
plt.xlabel("First principal component")
plt.ylabel("Second Principal Component")
```

Out[26]: Text(0, 0.5, 'Second Principal Component')



In [ ]:

Loading [MathJax]/extensions/Safe.js