In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import metrics
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

In [3]:
```
cancer.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [4]:
```
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
```

In [5]:
```
df = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
df.head()
```

Out[5]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.33	184.60	2
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.41	158.80	1
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.53	152.50	1
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.50	98.87	
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.67	152.20	1

5 rows × 30 columns

In [6]:
```
# scaling

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
scaled_data = scaler.transform(df)
```

In [9]:
```
scaled_data.shape
```

Out[9]: (569, 30)

In [10]:
```
from sklearn.decomposition import PCA
```

In [11]:
```
pca = PCA(n_components=5)
pca.fit(scaled_data)
```

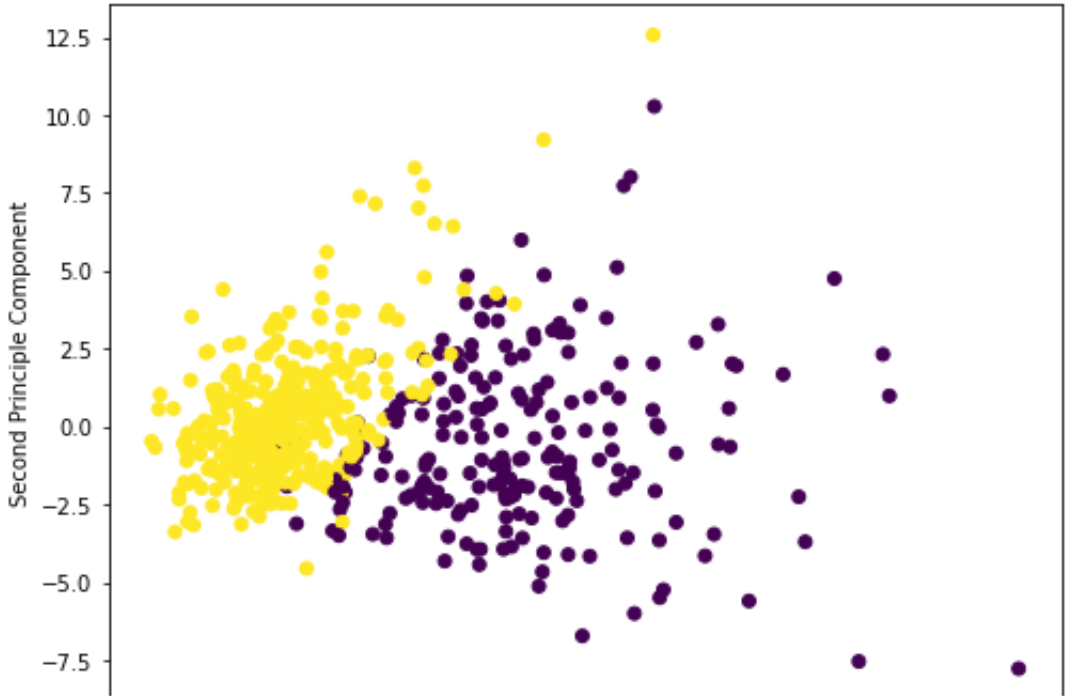
Out[11]: PCA(n_components=5)

In [12]:
```
# transform data
x_pca = pca.transform(scaled_data)
```

In [14]:
```
x_pca.shape
```

Out[14]: (569, 5)

In [17]:
```
plt.figure(figsize = (8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c = cancer['target'])
plt.xlabel("First Principle Component")
plt.ylabel("Second Principle Component")
plt.show()
```



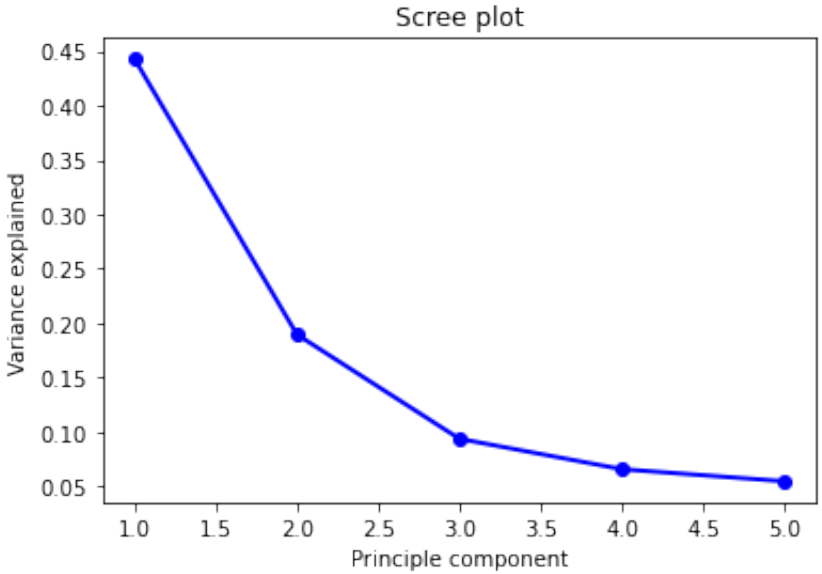
In [18]:
```
# pca components
pca.n_components_
```

Out[18]: 5

In [19]:
```
# explianed variance
pca.explained_variance_ratio_
```

Out[19]:
```
array([0.44272026, 0.18971182, 0.09393163, 0.06602135, 0.05495768])
```

In [23]:
```
# scree plot
pc_values = np.arange(pca.n_components_)+1
plt.plot(pc_values,pca.explained_variance_ratio_ ,'o-',linewidth =2,color = 'blue')
plt.title("Scree plot")
plt.xlabel("Principle component")
plt.ylabel("Variance explained")
plt.show()
```



In [ ]:
```
# from graph we will select 2 components,and by using we will build PCA again
```

In [24]:
```
pca_new = PCA(n_components=2)
pca_new.fit(scaled_data)
# transform data
x_pca_new = pca_new.transform(scaled_data)
x_pca_new.shape
```

Out[24]: (569, 2)

In [ ]:
```

```

In [ ]:
```

```