PCA_Data_Mining

In [16]:
```
import pandas as pd
import numpy as np

#here we are going to use inbuilt dataset
from sklearn.datasets import load_breast_cancer

#instantiating
cancer = load_breast_cancer(as_frame=True)
#creating dataframe
df = cancer.frame

#checking shape
print('original Dataframe Shape: ', df.shape)

#input features
X= df[cancer['feature_names']]
print('Input Datframe shape: ' ,X.shape )

```

```
original Dataframe Shape:  (569, 31)
Input Datframe shape:  (569, 30)
```

applying the first step to to standarize the data and for that we will calculate the mean and std of eacg feature in the feature space

In [18]:
```
#mean
X_mean =X.mean()

#std
X_std = X.std()

#standardaization]
Z=(X-X_mean) /X_std
```

The covariance matrix helps us to visualize the how strong dependency of two feature is with each other in the feature space

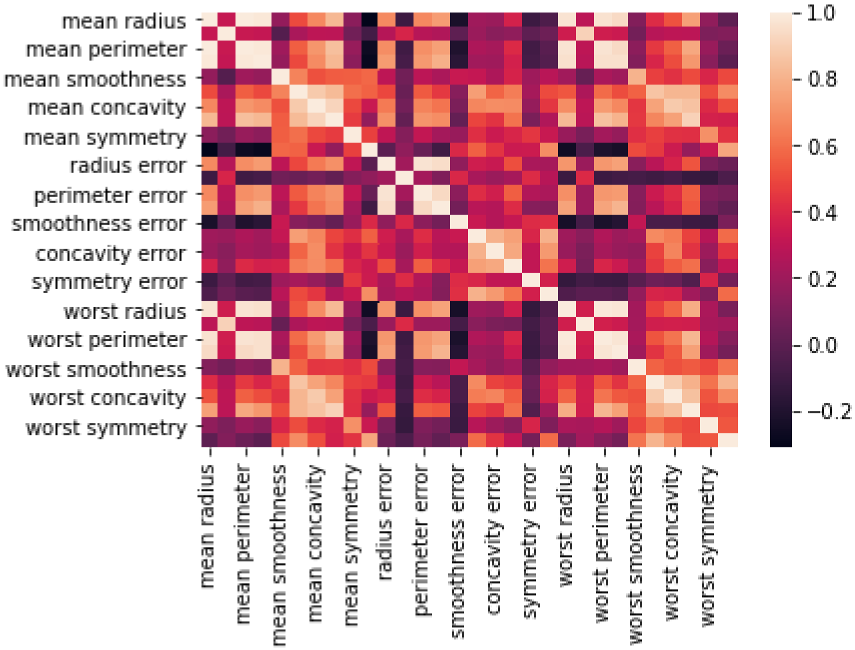
In [19]:

```
#covarince
c=Z.cov()

#plot rhe covarince matric

import matplotlib.pyplot as plt
import seaborn as sns

sns.heatmap(c)
plt.show()
```



In [20]:

```
#now we will try to compute egienval and eigen vect for out feature spa


eigenvalues , eigenvectors = np.linalg.eig(c)
print('Eigen values:\n ' ,eigenvalues)
print('Eigen values:\n ' ,eigenvalues.shape)
print('Eigen Vectors:\n ', eigenvectors.shape)
```

```
Eigen values:
  [1.32816077e+01 5.69135461e+00 2.81794898e+00 1.98064047e+00
 1.64873055e+00 1.20735661e+00 6.75220114e-01 4.76617140e-01
 4.16894812e-01 3.50693457e-01 2.93915696e-01 2.61161370e-01
 2.41357496e-01 1.57009724e-01 9.41349650e-02 7.98628010e-02
 5.93990378e-02 5.26187835e-02 4.94775918e-02 1.33044823e-04
 7.48803097e-04 1.58933787e-03 6.90046388e-03 8.17763986e-03
 1.54812714e-02 1.80550070e-02 2.43408378e-02 2.74394025e-02
 3.11594025e-02 2.99728939e-02]
Eigen values:
  (30,)
Eigen Vectors:
  (30, 30)
```

In [21]:
```
#sorting the eig val in descinding order and sort the corresponding eig

#index the eig val in desc ord
idx= eigenvalues.argsort()[::-1]

#sort in desc
eigenvalues = eigenvalues[idx]

#sort the correspondong eig vect accordingly

eigenvectors = eigenvectors[:,idx]


```

In [22]:
```
#explained var is the term that gives us an idea of the amount of the t
#which has been reatained by selectinf the principal compnentes instead

explained_var = np.cumsum(eigenvalues) / np.sum(eigenvalues)

explained_var
```

Out[22]:
```
array([0.44272026, 0.63243208, 0.72636371, 0.79238506, 0.84734274,
       0.88758796, 0.9100953 , 0.92598254, 0.93987903, 0.95156881,
       0.961366  , 0.97007138, 0.97811663, 0.98335029, 0.98648812,
       0.98915022, 0.99113018, 0.99288414, 0.9945334 , 0.99557204,
       0.99657114, 0.99748579, 0.99829715, 0.99889898, 0.99941502,
       0.99968761, 0.99991763, 0.99997061, 0.99999557, 1.        ])
```

In [23]:
```
#determine the number od principal compnents
#considering the num of principal comp of any nvalue our choice or by l
#consodereing the exp var more than equal to 50%

n_components  = np.argmax(explained_var >= 0.50 ) +1
n_components
```

Out[23]: 2

project the data onto the selected ptinceipal components

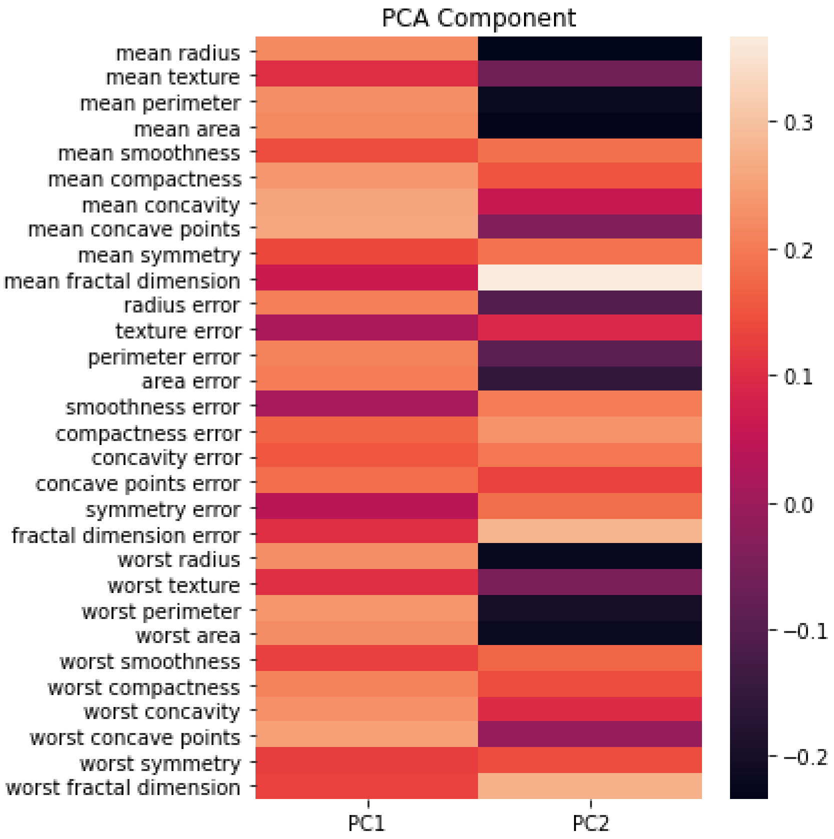
finding ht e projectin matrix, it is a matrix of egienvect correspondinf to the largest eig val of the cov matrix of the data. it projects the high-dimensinal dataset onto lower dimensional subspace

the eigenvect odf the cov matrix of the data are reffered toi as the principa asxes of the data, and the projection od the data instances onto these principal axes are called the principal cpmponents

In [26]:

```python
#pca compnenent or unit matrix
u = eigenvectors[:,:n_components]
pca_component = pd.DataFrame(u,
                             index = cancer['feature_names'],
                             columns = ['PC1' ,'PC2'])

#plot
plt.figure(figsize=(5,7))
sns.heatmap(pca_component)
plt.title('PCA Component')
plt.show()
```



In [29]:

```python
#matrix mult or dot product
Z_pca = Z @ pca_component
#rename the columns names
Z_pca.rename({'PC1': 'PCA1', 'PC2' : 'PCA2'} , axis = 1, inplace=True)
print(Z_pca)
```

```
          PCA1       PCA2
0     9.184755   1.946870
1     2.385703  -3.764859
2     5.728855  -1.074229
3     7.116691  10.266556
4     3.931842  -1.946359
..         ...        ...
564   6.433655  -3.573673
565   3.790048  -3.580897
566   1.255075  -1.900624
567  10.365673   1.670540
568  -5.470430  -0.670047

[569 rows x 2 columns]
```

In [31]:

```
#pca usinf sklearn

#importing pca

from sklearn.decomposition import PCA

#lets say componnets = 2

pca= PCA(n_components=2)
pca.fit(Z)
x_pca = pca.transform(Z)

#creating the dataframe

df_pca1 = pd.DataFrame(x_pca,
                      columns=['PC{}'.format(i+1)
                              for i in range(n_components)])

print(df_pca1)
```

```
           PC1        PC2
0     9.184755   1.946870
1     2.385703  -3.764859
2     5.728855  -1.074229
3     7.116691  10.266556
4     3.931842  -1.946359
..         ...        ...
564   6.433655  -3.573673
565   3.790048  -3.580897
566   1.255075  -1.900624
567  10.365673   1.670540
568  -5.470430  -0.670047

[569 rows x 2 columns]
```

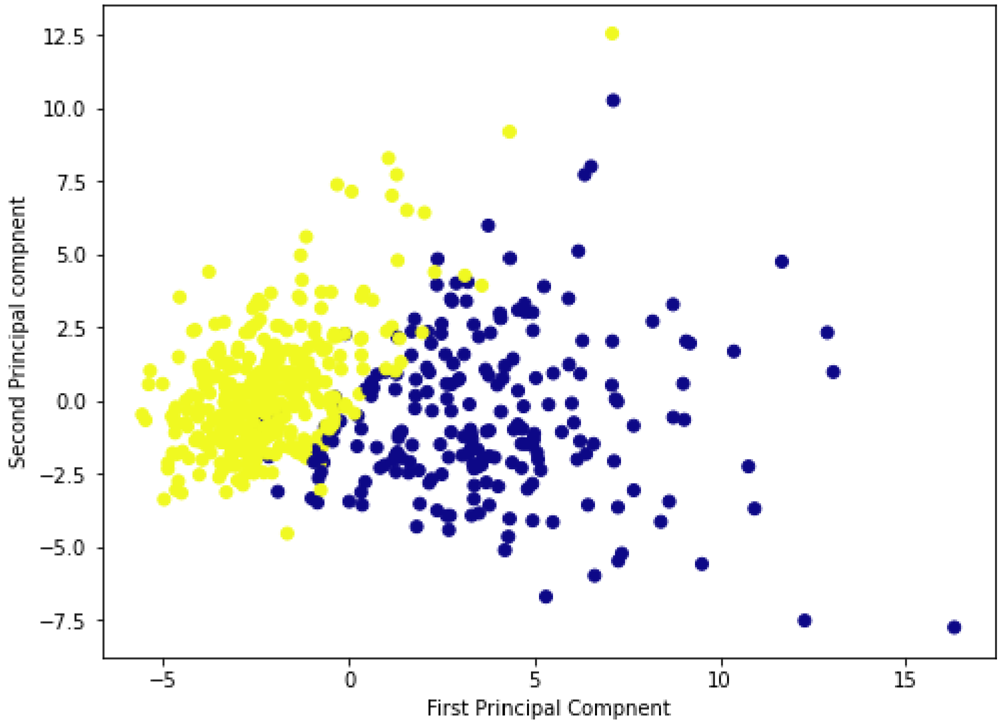
In [32]:
```python
#plotting

plt.figure(figsize=(8,6))

plt.scatter(x_pca[:,0], x_pca[:,1], c=cancer['target'] , cmap='plasma')

#labeling x and y

plt.xlabel('First Principal Compnent')
plt.ylabel('Second Principal compnent')
plt.show()
```



In [33]:
```python
pca.components_
```

Out[33]:
```
array([[ 0.21890244,  0.10372458,  0.22753729,  0.22099499,  0.14258969,
         0.23928535,  0.25840048,  0.26085376,  0.13816696,  0.06436335,
         0.20597878,  0.01742803,  0.21132592,  0.20286964,  0.01453145,
         0.17039345,  0.15358979,  0.1834174 ,  0.04249842,  0.10256832,
         0.22799663,  0.10446933,  0.23663968,  0.22487053,  0.12795256,
         0.21009588,  0.22876753,  0.25088597,  0.12290456,  0.13178394],
       [-0.23385713, -0.05970609, -0.21518136, -0.23107671,  0.18611302,
         0.15189161,  0.06016536, -0.0347675 ,  0.19034877,  0.36657547,
        -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
         0.2327159 ,  0.19720728,  0.13032156,  0.183848  ,  0.28009203,
        -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
         0.14359317,  0.09796411, -0.00825724,  0.14188335,  0.27533947]])
```

In [ ]:
```python

```