PCA project features onto principal components

Goal : reduce dimentionality while losing only a small amount of infromation

Import libs

In [1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
```

In [2]:
```
#### Load Data
```

In [3]:
```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

In [4]:
```
cancer.keys()
```

Out[4]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filen
ame'])
```

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

In [6]:
```
df.shape
```

Out[6]:
```
(569, 30)
```

In [7]:
```
(df.dtypes == 'object').any()
```

Out[7]:
```
False
```

In [8]:
```
df.head(5)
```

Out[8]:

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

5 rows × 30 columns

In [ ]:

Scale the data

In [9]:
```
from sklearn.preprocessing import StandardScaler
sc =  StandardScaler()
sc.fit(df)
sc_arr = sc.transform(df)
```

Create a pca object with the 2 components as a parameter

In [10]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(sc_arr)
x_pca = pca.transform(sc_arr)
```

In [ ]:

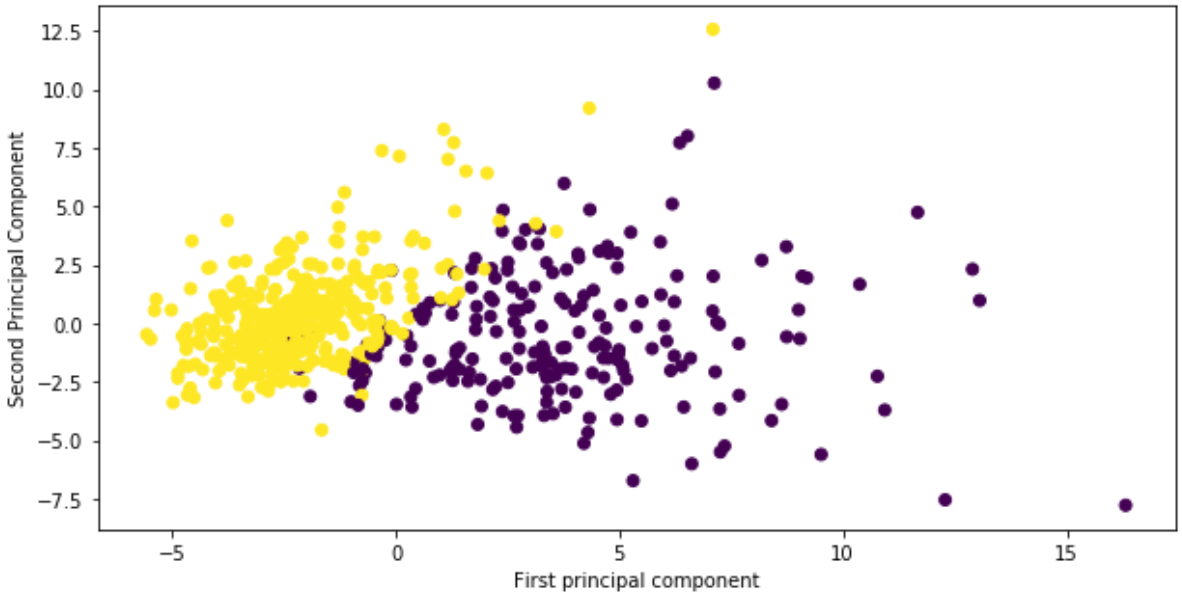
In [11]:
```
x_pca.shape
```

Out[11]:
```
(569, 2)
```

In [ ]:

In [12]:
```
plt.figure(figsize=(10,5))
plt.scatter(x_pca[:,0], x_pca[:,1], c = cancer['target'])
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[12]:
```
Text(0, 0.5, 'Second Principal Component')
```



In [ ]:

In [13]:
```
pca.components_
```

Out[13]:
```
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 [14]:
```
pca.components_.shape
```

Out[14]:
```
(2, 30)
```

In [ ]:

In [15]:
```
df_components = pd.DataFrame(data = pca.components_, columns=cancer['feature_names'])
df_components
```

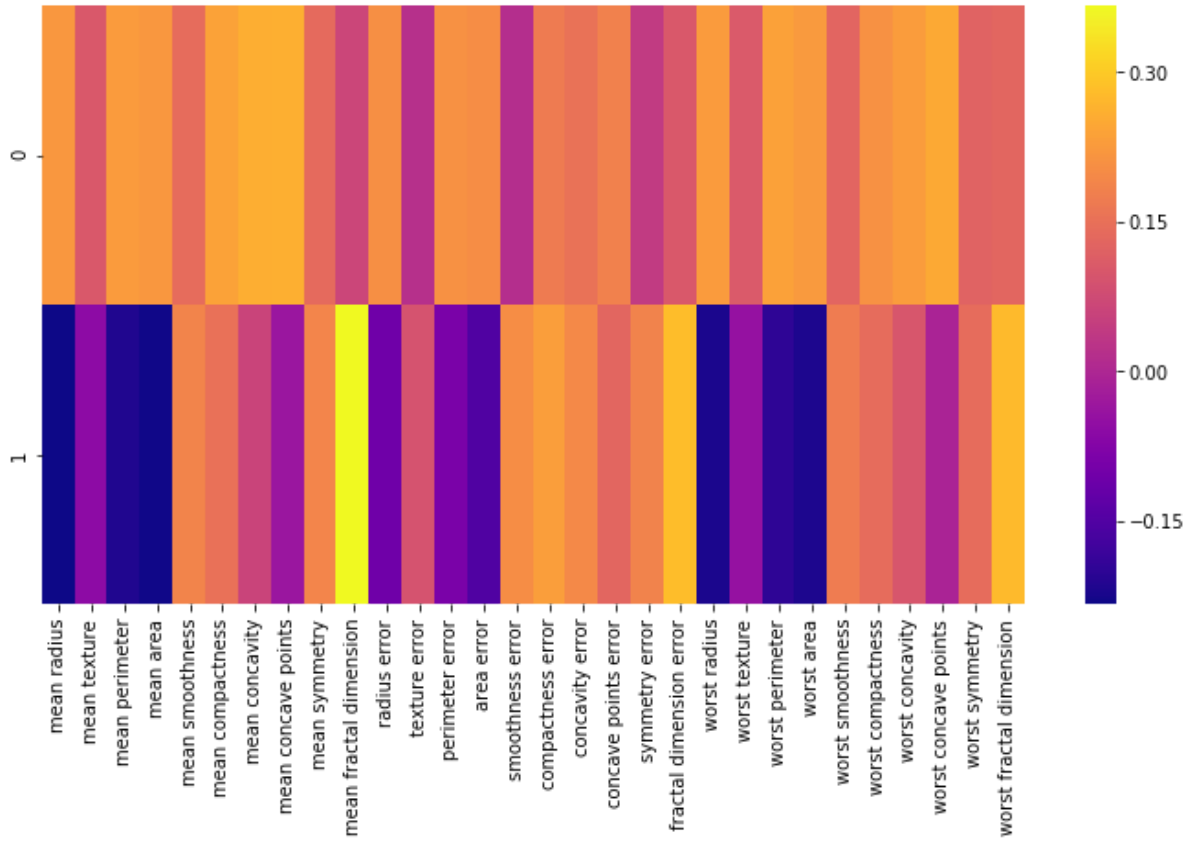
Out[15]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
0	0.218902	0.103725	0.227537	0.220995	0.142590	0.239285	0.258400	0.260854	
1	-0.233857	-0.059706	-0.215181	-0.231077	0.186113	0.151892	0.060165	-0.034768	

2 rows × 30 columns

In [16]:
```
plt.figure(figsize=(12,6))
sns.heatmap(df_components,cmap='plasma',yticklabels=True)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1b199d24f98>



In [ ]:
```
```

Create a PCA that will retain 99% of the variance

In [17]:
```
pca2 = PCA(n_components=0.99)
pca2.fit(sc_arr)
x_pca2 = pca2.transform(sc_arr)
```

In [18]:
```
x_pca2.shape
```

Out[18]: (569, 17)