```
from sklearn.datasets import load_breast_cancer
```

```
breast = load_breast_cancer()
```

```
breast
```

```
breast_data = breast.data
```

```
breast_data
```

```
breast_data.shape
```

```
breast_data
```

```
breast_labels = breast.target
```

```
breast_labels
```

```
breast_labels.shape
```

```
(569,)
```

```
import numpy as np

labels = np.reshape(breast_labels,(569,1))
```

```
labels
```

Show hidden output

```
final_breast_data = np.concatenate([breast_data,labels],axis=1)

final_breast_data.shape
```

```
(569, 31)
```

```
final_breast_data
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 4.601e-01, 1.189e-01,
        0.000e+00],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 2.750e-01, 8.902e-02,
        0.000e+00],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 3.613e-01, 8.758e-02,
        0.000e+00],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 2.218e-01, 7.820e-02,
        0.000e+00],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 4.087e-01, 1.240e-01,
        0.000e+00],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 2.871e-01, 7.039e-02,
        1.000e+00]])
```

```
import pandas as pd
breast_dataset = pd.DataFrame(final_breast_da
```

```
breast_dataset
```

	0	1	2	3	4	5	6	7	8	9	...	21	22	23
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	26.50	98.87	567.7
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2027.0
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1731.0
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1124.0
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1821.0
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	30.37	59.16	268.6

569 rows × 31 columns

```
final_breast_data
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 4.601e-01, 1.189e-01,
        0.000e+00],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 2.750e-01, 8.902e-02,
        0.000e+00],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 3.613e-01, 8.758e-02,
        0.000e+00],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 2.218e-01, 7.820e-02,
        0.000e+00],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 4.087e-01, 1.240e-01,
        0.000e+00],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 2.871e-01, 7.039e-02,
        1.000e+00]])
```

```
features = breast.feature_names
```

```
features
```

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
features_labels = np.append(features,'label')
```

```
breast_dataset.columns = features_labels
```

```
breast_dataset.head()
```

	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 × 31 columns

```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)
```

```
<ipython-input-23-2c40a603cf26>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or S
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=T


  breast_dataset['label'].replace(0, 'Benign',inplace=True)
```

```
breast_dataset.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	.
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	

5 rows × 31 columns

```
from sklearn.preprocessing import StandardScaler
x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x) # normalizing the features
```

```
x.shape
```

```
(569, 30)
```

```
np.mean(x),np.std(x)
```

```
(-6.826538293184326e-17, 1.0)
```

```
feat_cols = ['feature'+str(i) for i in range(x.shape[1])]
```

```
normalised_breast = pd.DataFrame(x,columns=feat_cols)
```

```
normalised_breast.head()
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	...
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919	...
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450	...

5 rows × 30 columns

```
normalised_breast.tail()
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6	feature7	feature8	feature9	.
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285	2.320965	-0.312589	-0.931027	
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043	1.263669	-0.217664	-1.058611	
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588	0.105777	-0.809117	-0.895587	
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944	2.658866	2.137194	1.043695	
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873	-1.261820	-0.820070	-0.561032	

5 rows × 30 columns

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)
```

```
principal_breast_Df = pd.DataFrame(data = principalComponents_breast, columns = ['principal component 1',
```

```
principal_breast_Df.tail()
```

	principal component 1	principal component 2
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

```
principal_breast_Df
```

	principal component 1	principal component 2
0	9.192837	1.948583
1	2.387802	-3.768172
2	5.733896	-1.075174
3	7.122953	10.275589
4	3.935302	-1.948072
...	...	...
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

569 rows × 2 columns

```
print('Explained variation per principal component: {}'.format(pca_breast.explained_variance_ratio_))
```

```
Explained variation per principal component: [0.44272026 0.18971182]
```

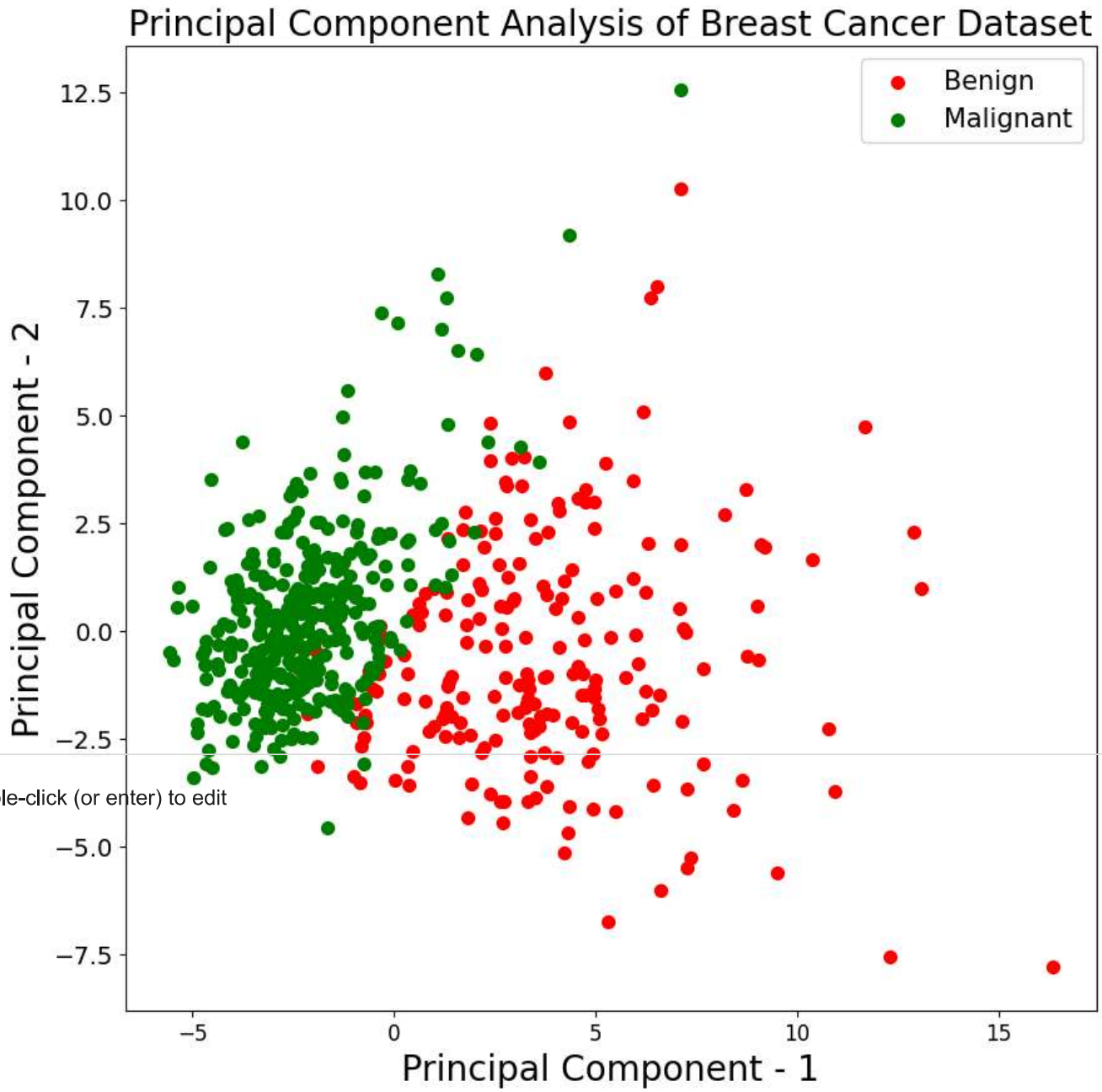
```
import matplotlib.pyplot as plt
```

```
plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)

plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer Dataset",fontsize=20)
targets = ['Benign', 'Malignant']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = breast_dataset['label'] == target
    plt.scatter(principal_breast_Df.loc[indicesToKeep, 'principal component 1']
               , principal_breast_Df.loc[indicesToKeep, 'principal component 2'], c = color, s = 50)

plt.legend(targets,prop={'size': 15})
```

```
<matplotlib.legend.Legend at 0x7f560a04fee0>
<Figure size 640x480 with 0 Axes>
```



Double-click (or enter) to edit