

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
```

```
#load dataset
b = load_breast_cancer()
bdata = b.data
print(bdata)
print(bdata.shape)
```

```
[[1.799e+01 1.038e+01 1.228e+02 ... 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 ... 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 ... 2.430e-01 3.613e-01 8.758e-02]
 ...
 [1.660e+01 2.808e+01 1.083e+02 ... 1.418e-01 2.218e-01 7.820e-02]
 [2.060e+01 2.933e+01 1.401e+02 ... 2.650e-01 4.087e-01 1.240e-01]
 [7.760e+00 2.454e+01 4.792e+01 ... 0.000e+00 2.871e-01 7.039e-02]]
(569, 30)
```

```
#use of target
blabels = b.target
print(blabels)
print(blabels.shape)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

```
#reshaping the dataset by adding label to it
labels=np.reshape(blabels, (569,1))
final=np.concatenate([bdata, labels], axis=1)
print(final.shape)

(569, 31)
```

```
#create a dataframe
bdataset = pd.DataFrame(final)
print(bdataset)
```

```
   0    1    2    3    4    5    6    7    8  \
0  17.99  10.38  122.80  1001.0  0.11840  0.27760  0.30010  0.14710  0.2419
1  20.57  17.77  132.90  1326.0  0.08474  0.07864  0.08690  0.07017  0.1812
2  19.69  21.25  130.00  1203.0  0.10960  0.15990  0.19740  0.12790  0.2069
3  11.42  20.38  77.58  386.1  0.14250  0.28390  0.24140  0.10520  0.2597
4  20.29  14.34  135.10  1297.0  0.10030  0.13280  0.19800  0.10430  0.1809
..  ...  ...  ...  ...  ...  ...  ...  ...  ...
564 21.56  22.39  142.00  1479.0  0.11100  0.11590  0.24390  0.13890  0.1726
565 20.13  28.25  131.20  1261.0  0.09780  0.10340  0.14400  0.09791  0.1752
566 16.60  28.08  108.30  858.1  0.08455  0.10230  0.09251  0.05302  0.1590
567 20.60  29.33  140.10  1265.0  0.11780  0.27700  0.35140  0.15200  0.2397
568 7.76  24.54  47.92  181.0  0.05263  0.04362  0.00000  0.00000  0.1587

   9  ...  21  22  23  24  25  26  27  \
0  0.07871  ...  17.33  184.60  2019.0  0.16220  0.66560  0.7119  0.2654
1  0.05667  ...  23.41  158.80  1956.0  0.12380  0.18660  0.2416  0.1860
2  0.05999  ...  25.53  152.50  1709.0  0.14440  0.42450  0.4504  0.2430
3  0.09744  ...  26.50  98.87  567.7  0.20980  0.86630  0.6869  0.2575
4  0.05883  ...  16.67  152.20  1575.0  0.13740  0.20500  0.4000  0.1625
..  ...  ...  ...  ...  ...  ...  ...  ...
564 0.05623  ...  26.40  166.10  2027.0  0.14100  0.21130  0.4107  0.2216
565 0.05533  ...  38.25  155.00  1731.0  0.11660  0.19220  0.3215  0.1628
566 0.05648  ...  34.12  126.70  1124.0  0.11390  0.30940  0.3403  0.1418
567 0.07016  ...  39.42  184.60  1821.0  0.16500  0.86810  0.9387  0.2650
568 0.05884  ...  30.37  59.16  268.6  0.08996  0.06444  0.0000  0.0000

   28  29  30
0  0.4601  0.11890  0.0
1  0.2750  0.08902  0.0
2  0.3613  0.08758  0.0
3  0.6638  0.17300  0.0
4  0.2364  0.07678  0.0
..  ...  ...  ...
564 0.2060  0.07115  0.0
565 0.2572  0.06637  0.0
566 0.2218  0.07820  0.0
567 0.4087  0.12400  0.0
568 0.2871  0.07039  1.0

[569 rows x 31 columns]
```

```
# print the features
f = b.feature_names
print(f)

['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']

#add the missing labels to it
feature_labels = np.append(f, "label")

#embedding the column names in dataframe
bdataset.columns = feature_labels
bdataset.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0.6656	0.7119
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0.1866	0.2416
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0.4245	0.4504
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0.8663	0.6869
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0.2050	0.4000

5 rows × 31 columns

```
#replace the data with values of benign and malignant using .replace function
bdataset["label"].replace(0, 'Benign', inplace=True)
bdataset["label"].replace(1, 'Malignant', inplace=True)
bdataset.tail()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity
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	0.14100	0.21130	0.4100
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	0.11660	0.19220	0.3210
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	0.11390	0.30940	0.3400
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	0.16500	0.86810	0.9380
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	0.08996	0.06444	0.0000

5 rows × 31 columns

PCA standardisation

```
from sklearn.preprocessing import StandardScaler
x=bdataset.loc[:, f].values
x=StandardScaler().fit_transform(x)
print(x.shape)
```

(569, 30)

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

1.0
-6.118909323768877e-16

```
#convert normalised features into tabular form
feat_cols=['feature'+str(i) for i in range(x.shape[1])]
n = pd.DataFrame(x, columns=feat_cols)
print(n)
```

	feature0	feature1	feature2	feature3	feature4	feature5	feature6 \
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011
..
564	2.110995	0.721473	2.060786	2.343856	1.041842	0.219060	1.947285
565	1.704854	2.085134	1.615931	1.723842	0.102458	-0.017833	0.693043
566	0.702284	2.045574	0.672676	0.577953	-0.840484	-0.038680	0.046588
567	1.838341	2.336457	1.982524	1.735218	1.525767	3.272144	3.296944
568	-1.808401	1.221792	-1.814389	-1.347789	-3.112085	-1.150752	-1.114873
..
feature7	feature8	feature9	...	feature20	feature21	feature22 \	
0	2.532475	2.217515	2.255747	...	1.886690	-1.359293	2.303601
1	0.548144	0.001392	-0.868652	...	1.805927	-0.369203	1.535126
2	2.037231	0.939685	-0.398008	...	1.511870	-0.023974	1.347475
3	1.451707	2.867383	4.910919	...	-0.281464	0.133984	-0.249939
4	1.428493	-0.009560	-0.562450	...	1.298575	-1.466770	1.338539
..
564	2.320965	-0.312589	-0.931027	...	1.901185	0.117700	1.752563
565	1.263669	-0.217664	-1.058611	...	1.536720	2.047399	1.421940
566	0.105777	-0.809117	-0.895587	...	0.561361	1.374854	0.579001
567	2.658866	2.137194	1.043695	...	1.961239	2.237926	2.303601
568	-1.261820	-0.820070	-0.561032	...	-1.410893	0.764190	-1.432735
..
feature23	feature24	feature25	feature26	feature27	feature28 \		
0	2.001237	1.307686	2.616665	2.109526	2.296076	2.750622	
1	1.890489	-0.375612	-0.430444	-0.146749	1.087084	-0.243890	
2	1.456285	0.527407	1.082932	0.854974	1.955000	1.152255	
3	-0.550021	3.394275	3.893397	1.989588	2.175786	6.046041	
4	1.220724	0.220556	-0.313395	0.613179	0.729259	-0.868353	
..	
564	2.015301	0.378365	-0.273318	0.664512	1.629151	-1.360158	
565	1.494959	-0.691230	-0.394820	0.236573	0.733827	-0.531855	
566	0.427906	-0.809587	0.350735	0.326767	0.414069	-1.104549	
567	1.653171	1.430427	3.904848	3.197605	2.289985	1.919083	

```
568 -1.075813 -1.859019 -1.207552 -1.305831 -1.745063 -0.048138
```

```
feature29
0 1.937015
1 0.281190
2 0.201391
3 4.935010
4 -0.397100
.. ...
564 -0.709091
565 -0.973978
566 -0.318409
567 2.219635
568 -0.751207

[569 rows x 30 columns]
```

```
#projecting 30 dim(cols) space to 2D
from sklearn.decomposition import PCA
p = PCA(n_components=2)
pc_breast = p.fit_transform(x)
breast=pd.DataFrame(data=pc_breast, columns=['PC1', 'PC2'])
breast.tail()
```

	PC1	PC2
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

```
explained_variance_ratio_ = pc_breast.explained_variance_ratio_
print("explained variation per principal component: {}".format(explained_variance_ratio_))
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-59-6fc18cc1afde> in <cell line: 1>()
----> 1 explained_variance_ratio_ = pc_breast.explained_variance_ratio_
      2 print("explained variation per principal component: {}".format(explained_variance_ratio_))

AttributeError: 'numpy.ndarray' object has no attribute 'explained_variance_ratio_'
```

```
#PLOTING
import matplotlib.pyplot as plt
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('PC1', fontsize=20)
plt.ylabel('PC2', fontsize=20)
plt.title('PCA of breast cancer dataset', fontsize=20)
targets=['Benign', 'Malignant']
colors=['r','g']
for targets, colors in zip(targets, colors):
    indicesToKeep = bdataset['label']==targets
    plt.scatter(breast.loc[indicesToKeep, 'PC1'], breast.loc[indicesToKeep, 'PC2'], c=colors, s=50 )
plt.legend(targets, prop={'size' :15})
plt.show()
```

PCA of breast cancer dataset

