In [26]:
```
#importing the necessary packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [27]:
```
#from the sample datasets of sklearn importing the breastcancer dataset

from sklearn.datasets import load_breast_cancer
```

In [28]:
```
#creating an instance for the breast_cancer dataset
cancer= load_breast_cancer()
```

In [29]:
```
#checking out the keyvalues of our dataset

print(cancer.keys())
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

In [30]:
```python
#checking out the Description value in our cancer instance of our breast_cancer dataset
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)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radius, field
        13 is Radius SE, field 23 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994)
     163-171.
```

In [33]:
```
#analysing the above description gives us much of the information on shape, size and cla
ss in our dataset

#time to use pandas DataFrame function to create our dataset in the form of a dataframe
#this dataframe will have all the data of this breastcancer dataset and the column names
#pulled from the feature names of this sample dataset

df1= pd.DataFrame(cancer['data'], columns=cancer['feature_names'])

df1.head()
```

Out[33]:

	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 × 30 columns

In [35]:
```
#Lets check out the shape of our df1
df1.shape
```

Out[35]:
```
(569, 30)
```

In [ ]:
```
#we can see that we have 30 features.

#our task here is to minimize the dimensionality of our data from 30 to a less complex s
ay 2D dataset
# as high dimensionaility leads to underfitting and  thus to inaccuracies

# The first and foremost step while performing Principal component analysis to reduce th
e dimensionality
#is to bring all our attributes currently under different scales to a single scale.
# this can be done using Standard Scaling or min max scaling using sklearn

#As we can see in our data set the mean perimeter attribute
#has altogether a different scale and span if compared to the mean compactness.

#So, because we are planning to create a new vector space with all of these dimensions,
#we must scale down or scale up the values, to bring their span in a close range
```

In [41]:
```
from sklearn.preprocessing import StandardScaler
scaler= StandardScaler()

scaler.fit(df1)
scaleddata=scaler.transform(df1)
```

scaling could have been donw using minmax scaling as well

from sklearn.preprocessing import MinMaxScaler

scaler1= MinMaxScaler()

scaler1.fit(df1) scaleddata1= scaler1.transform(df1)

In [42]:
```
scaleddata
```

Out[42]:
```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

In [43]:
```
#now lets apply PCA to reduce our dimesionality

from sklearn.decomposition import PCA

pcaobject= PCA(n_components=2)
```

In [44]:
```
pcaobject.fit(scaleddata)
```

Out[44]:
```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=None,
    svd_solver='auto', tol=0.0, whiten=False)
```

In [45]:
```
Xi=pcaobject.transform(scaleddata)
```

In [46]:
```
Xi.shape
```

Out[46]:
```
(569, 2)
```

In [47]:
```
scaleddata.shape
```

Out[47]:
```
(569, 30)
```

In [48]:
```
#our data set is ready right now as Xi and has 2 columsn
print(Xi)
```

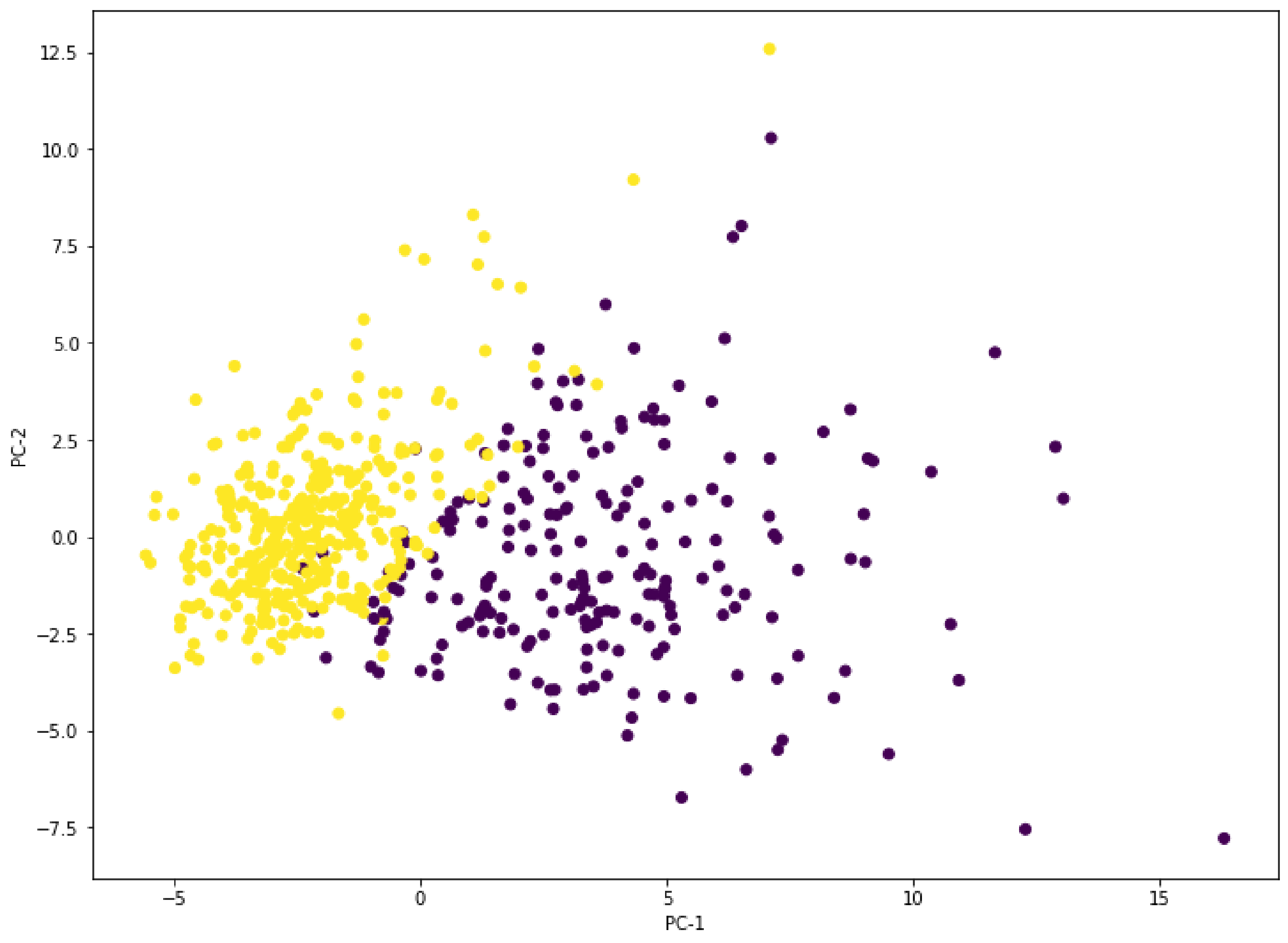
```
[[ 9.19283683  1.94858307]
 [ 2.3878018  -3.76817174]
 [ 5.73389628 -1.0751738 ]
 ...
 [ 1.25617928 -1.90229671]
 [10.37479406  1.67201011]
 [-5.4752433  -0.67063679]]
```

In [67]:
```
# Just to quickly visualise our data set

plt.figure(figsize=(12,9))
plt.scatter(Xi[:,0], Xi[:,1], c=cancer.target)

plt.xlabel('PC-1')
plt.ylabel('PC-2')
```

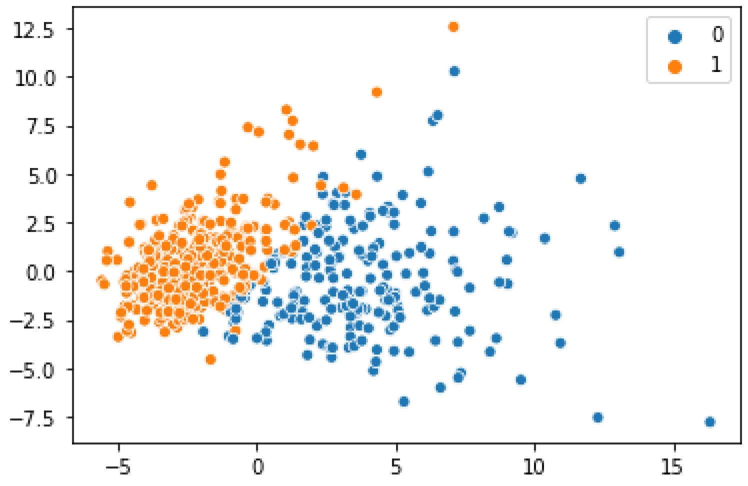
Out[67]:
```
Text(0, 0.5, 'PC-2')
```



```
In [70]: import seaborn as sns

         sns.scatterplot(Xi[:,0], Xi[:,1], hue=cancer.target )
```

```
Out[70]: <matplotlib.axes._subplots.AxesSubplot at 0x58d5151248>
```



```
In [73]: print(cancer.target_names)
```

```
['malignant' 'benign']
```