Import Libraries

```
# import all libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Loading data

```
#import the breast _cancer dataset
from sklearn.datasets import load_breast_cancer
data=load_breast_cancer()
data.keys()

# Check the output classes
print(data['target_names'])

# Check the input attributes
print(data['feature_names'])
```

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

Apply PCA

```
# construct a dataframe using pandas
df1=pd.DataFrame(data['data'],columns=data['feature_names'])

# Scale data before applying PCA
scaling=StandardScaler()

# Use fit and transform method
scaling.fit(df1)
Scaled_data=scaling.transform(df1)

# Set the n_components=3
principal=PCA(n_components=3)
principal.fit(Scaled_data)
x=principal.transform(Scaled_data)

```

```
# Check the dimensions of data after PCA
print(x.shape)
```

```
(569, 3)
```

check components

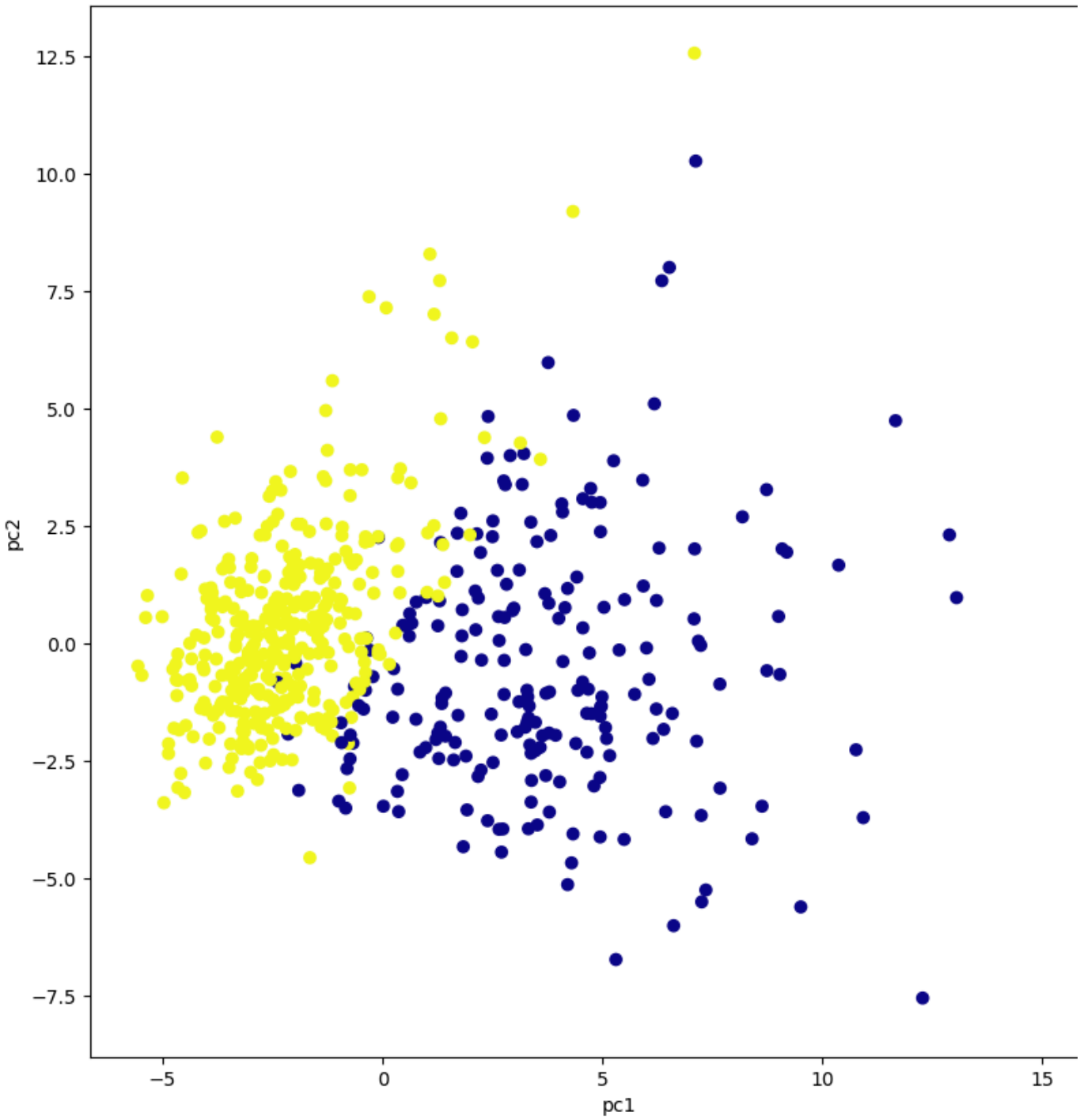
```
# Check the values of eigen vectors
# prodeced by principal components
principal.components_
```

```
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,  0.18611301,
         0.15189161,  0.06016536, -0.0347675 ,  0.19034876,  0.36657548,
        -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443045,
         0.23271588,  0.19720727,  0.13032158,  0.183848  ,  0.28009203,
        -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
         0.14359317,  0.09796411, -0.00825722,  0.14188335,  0.27533948],
       [-0.00853112,  0.0645498 , -0.00931409,  0.02869965, -0.1042913 ,
        -0.07409143,  0.00273402, -0.02556382, -0.04023954, -0.02257476,
         0.26848133,  0.37463384,  0.26664523,  0.21600677,  0.30883894,
         0.15478051,  0.17646417,  0.22465652,  0.28858403,  0.21150374,
        -0.0475071 , -0.04229787, -0.0485466 , -0.01190243, -0.25979754,
        -0.23607525, -0.17305703, -0.17034481, -0.27131287, -0.23279173]])
```

plot visualization

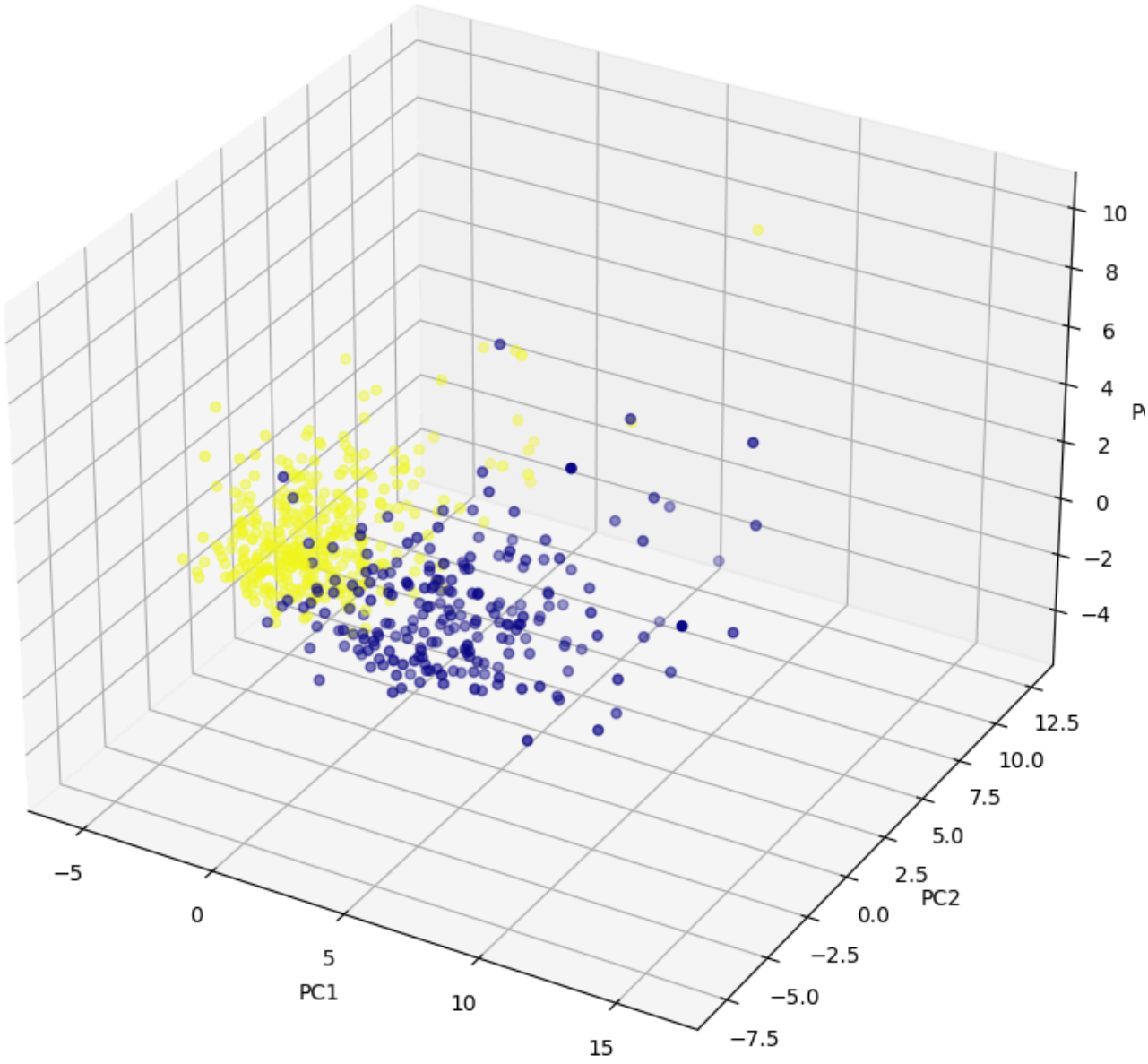
```
plt.figure(figsize=(10,10))
plt.scatter(x[:,0],x[:,1],c=data['target'],cmap='plasma')
plt.xlabel('pc1')
plt.ylabel('pc2')
```

```
Text(0, 0.5, 'pc2')
```




```
 1 # import relevant libraries for 3d graph
 2 from mpl_toolkits.mplot3d import Axes3D
 3 fig = plt.figure(figsize=(10,10))
 4
 5 # choose projection 3d for creating a 3d graph
 6 axis = fig.add_subplot(111, projection='3d')
 7
 8 # x[:,0]is pc1,x[:,1] is pc2 while x[:,2] is pc3
 9 axis.scatter(x[:,0],x[:,1],x[:,2], c=data['target'],cmap='plasma')
10 axis.set_xlabel("PC1", fontsize=10)
11 axis.set_ylabel("PC2", fontsize=10)
12 axis.set_zlabel("PC3", fontsize=10)
```

```
Text(0.5, 0, 'PC3')
```




```
1 # check how much variance is explained by each principal component
2 print(principal.explained_variance_ratio_)
```

```
[0.44272026 0.18971182 0.09393163]
```