Assignment 7

Title: Implement principle components analysis algorithm.

Theory:

Implementing PCA in Python with scikit-learn

In this practical, we will learn ~~about~~ about PCA (Principal Component Analysis) in python with scikit-learn. Let's start our learning step by step.

Why PCA?

- When there are many input attributes, it is difficult to visualize the data. There is a very famous term 'Curse of dimensionality' in the machine learning domain.
- Basically, it refers to the fact that a higher no. of attributes in a dataset adversely affects the accuracy & training time of the machine learning models.
- Principal Components Analysis (PCA) is a way to address this issue & is used for better data visualization & improving accuracy.

How does PCA works:

- PCA is an unsupervised pre-process task that is carried out before applying any ML algorithm. PCA is based on "orthogonal linear transformation" ~~ch~~ which is a mathematical technique to project the attributes of data set onto a new coordinate system. The attribute which describes the most variance is called the first principal component & is placed at the first coordinate.
- Similarly, the attribute which stands second in describing variance is called a second principle component & so on. In short, the complete dataset can be expressed in terms of principle components. Usually, more than 90% of the variance is explained by two/three principal components.
- Principle component analysis, or PCA, thus converts data from high dimensional space to low dimensional space by selecting the most important attributes that capture maximum information about the dataset.

Python Implementation

- To implement PCA in Scikit learn, it is essential to stadardize /normalize the data before applying PCA.
- PCA is imported from Sklearn. decomposition. We need to select the required no of principle components.
- Usually, n- components is choose to be 2 for better visualization but it matters & depends on data.
- By the fit and tranform method, the attributes are passed.
- The values of principal components can be checked using components while the variance explained by each principal components can be calculated using explained - Variance - ratio.

1. Import all the libraries
   # import all libraries
2. Loading data
   Load the breast - cancer dataset from sklearn datasets
3. Apply PCA
   - Standardize the dataset prior to PCA
   - Import PCA from Sklearn, decomposition
   - Choose the no. of principal components
4. Check components
   The principal components provide an array in which the no of row tells the no of principal components while the no of columns is equal to the no of feature in actual dat
5. Plot the components (visualization)
   Plot the principal components for better data visualation.
6. Calculate variance ratio
   Explained variance ratio provides an idea of how much variation is explained by principal components.

Conclusion - Thus, we have implemented principal components analysis algorithm

Import all the libraries

In [1]:
```
# 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

In [2]:
```
#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

In [3]:
```
# 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

In [4]:
```
# Check the values of eigen vectors
# prodeced by principal components
```

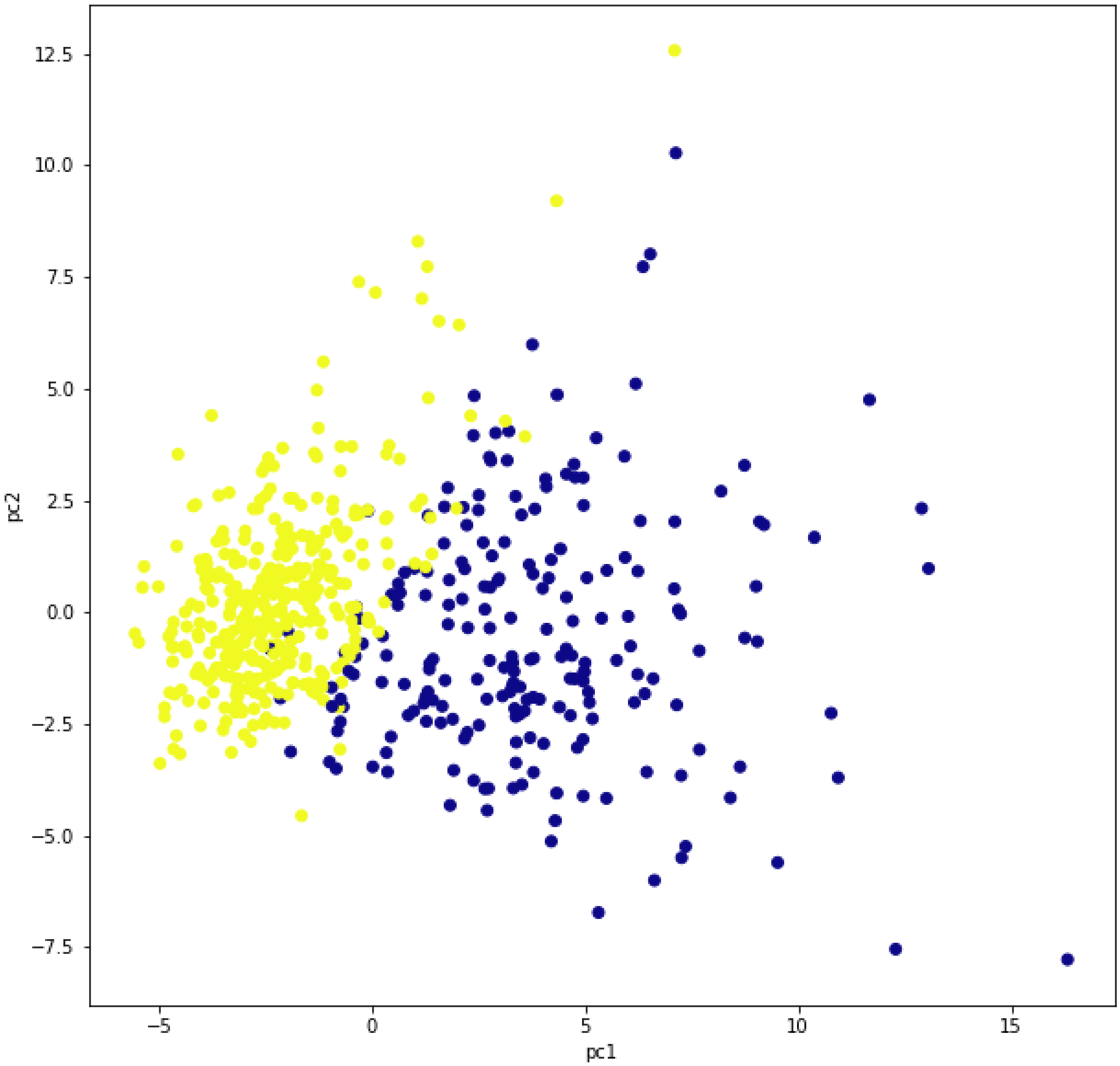
```
principal.components_
```

Out[4]:
```
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,  0.06016537, -0.03476749,  0.19034877,  0.36657548,
         -0.10555215,  0.08997968, -0.08945723, -0.15229263,  0.20443046,
          0.23271589,  0.19720728,  0.13032156,  0.183848  ,  0.28009202,
         -0.21986638, -0.0454673 , -0.19987843, -0.21935186,  0.17230435,
          0.14359317,  0.09796412, -0.00825723,  0.14188335,  0.27533947],
       [-0.00853124,  0.06454989, -0.00931422,  0.02869953, -0.10429189,
         -0.07409157,  0.00273382, -0.02556356, -0.04023994, -0.02257409,
          0.26848138,  0.37463367,  0.26664536,  0.21600653,  0.30883897,
          0.15477974,  0.17646376,  0.22465756,  0.28858429,  0.21150376,
         -0.04750698, -0.04229782, -0.0485465 , -0.01190231, -0.2597976 ,
         -0.23607562, -0.17305734, -0.17034409, -0.27131263, -0.23279132]])
```

Plot the components (Visualization)

In [5]:
```
plt.figure(figsize=(10,10))
plt.scatter(x[:,0],x[:,1],c=data['target'],cmap='plasma')
plt.xlabel('pc1')
plt.ylabel('pc2')
```

Out[5]:
```
Text(0, 0.5, 'pc2')
```

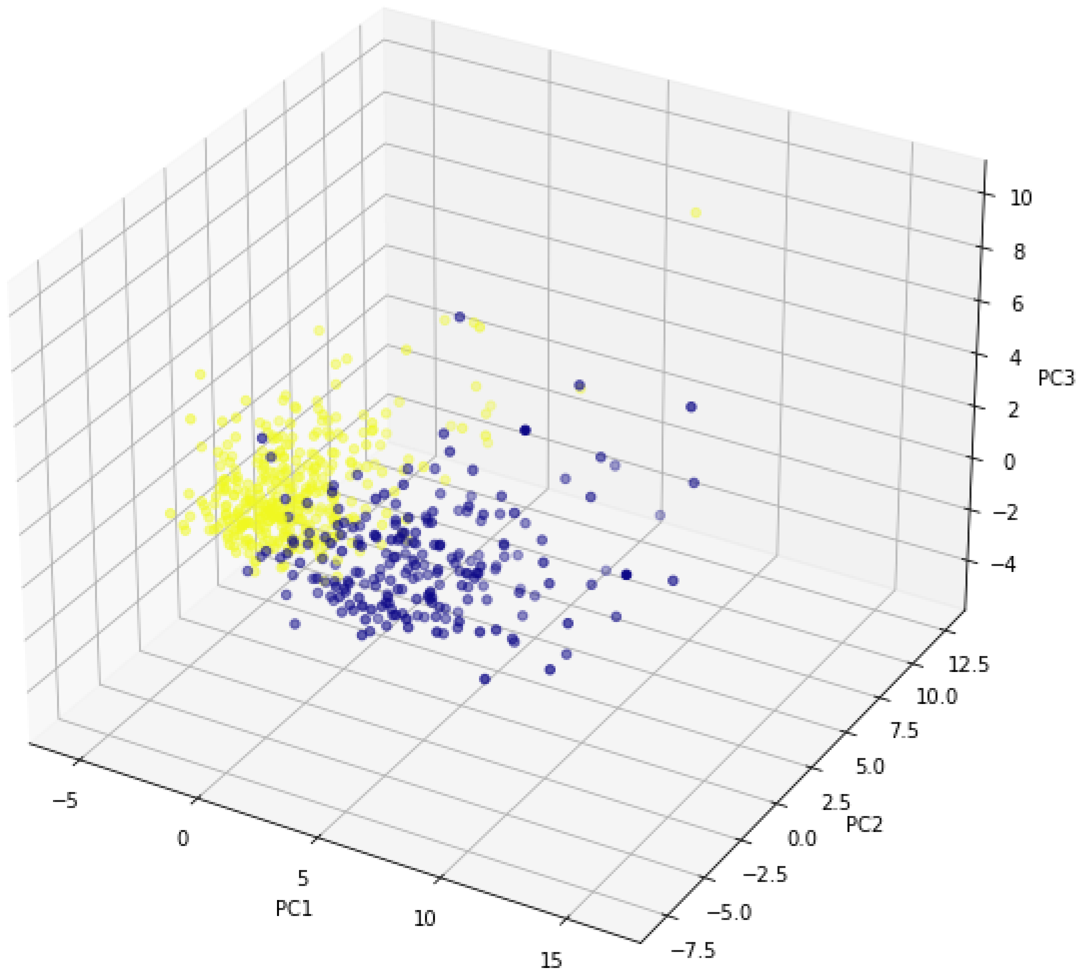


In [6]:
```python
# import relevant libraries for 3d graph
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(10,10))

# choose projection 3d for creating a 3d graph
axis = fig.add_subplot(111, projection='3d')

# x[:,0]is pc1,x[:,1] is pc2 while x[:,2] is pc3
axis.scatter(x[:,0],x[:,1],x[:,2], c=data['target'],cmap='plasma')
axis.set_xlabel("PC1", fontsize=10)
axis.set_ylabel("PC2", fontsize=10)
axis.set_zlabel("PC3", fontsize=10)
```

Out[6]: `Text(0.5, 0, 'PC3')`



Calculate variance ratio

In [7]:
```
# check how much variance is explained by each principal component
print(principal.explained_variance_ratio_)
```

```
[0.44272026 0.18971182 0.09393163]
```

In [ ]: