In [15]:

```
#Importing required libraries
import numpy as np
```

In [2]:

```
#Generate a dummy dataset.
#X = np.random.randint(10,50,100).reshape(20,5)
X=np.array([[2,1],[3,4],[5,0],[7,6],[9,2]])
# mean Centering the data
X_meaned = X - np.mean(X , axis = 0)
print(X)
print(X_meaned)
```

```
[[2 1]
 [3 4]
 [5 0]
 [7 6]
 [9 2]]
[[-3.2 -1.6]
 [-2.2  1.4]
 [-0.2 -2.6]
 [ 1.8  3.4]
 [ 3.8 -0.6]]
```

In [3]:

```
# calculating the covariance matrix of the mean-centered data.
cov_mat = np.cov(X_meaned , rowvar = False)
print(cov_mat)
```

```
[[8.2 1.6]
 [1.6 5.8]]
```

In [4]:

```
#Calculating Eigenvalues and Eigenvectors of the covariance matrix
eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)

print(eigen_values)
print(eigen_vectors)
```

```
[5. 9.]
[[ 0.4472136  -0.89442719]
 [-0.89442719 -0.4472136 ]]
```

In [5]:

```
#sort the eigenvalues in descending order
sorted_index = np.argsort(eigen_values)[::-1]

sorted_eigenvalue = eigen_values[sorted_index]
#similarly sort the eigenvectors
sorted_eigenvectors = eigen_vectors[:,sorted_index]
print(sorted_index)
print(sorted_eigenvalue)
print(sorted_eigenvectors)
```

```
[1 0]
[9. 5.]
[[-0.89442719  0.4472136 ]
 [-0.4472136  -0.89442719]]
```

In [6]:

```
# select the first n eigenvectors, n is desired dimension
# of our final reduced data.

n_components = 1 #you can select any number of components.
eigenvector_subset = sorted_eigenvectors[:,0:n_components]
print(eigenvector_subset)
```

```
[[-0.89442719]
 [-0.4472136 ]]
```

In [7]:

```
#Transform the data
X_reduced = np.dot(eigenvector_subset.transpose(),X_meaned.transpose()).transpose()
print(X_reduced)
```

```
[[ 3.57770876]
 [ 1.34164079]
 [ 1.34164079]
 [-3.13049517]
 [-3.13049517]]
```

In [13]:

```
import numpy as np
def PCA(X , num_components):

    #Step-1
    X_meaned = X - np.mean(X , axis = 0)

    #Step-2
    cov_mat = np.cov(X_meaned , rowvar = False)

    #Step-3
    eigen_values , eigen_vectors = np.linalg.eigh(cov_mat)

    #Step-4
    sorted_index = np.argsort(eigen_values)[::-1]
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:,sorted_index]

    #Step-5
    eigenvector_subset = sorted_eigenvectors[:,0:num_components]

    #Step-6
    X_reduced = np.dot(eigenvector_subset.transpose() , X_meaned.transpose() ).transpose

    return X_reduced

import pandas as pd

#Get the IRIS dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
data = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width'

#prepare the data
x = data.iloc[:,0:4]

#prepare the target
target = data.iloc[:,4]

#Applying it to PCA function
mat_reduced = PCA(x , 2)

#Creating a Pandas DataFrame of reduced Dataset
principal_df = pd.DataFrame(mat_reduced , columns = ['PC1', 'PC2'])

#Concat it with target variable to create a complete Dataset
principal_df = pd.concat([principal_df , pd.DataFrame(target)] , axis = 1)
```

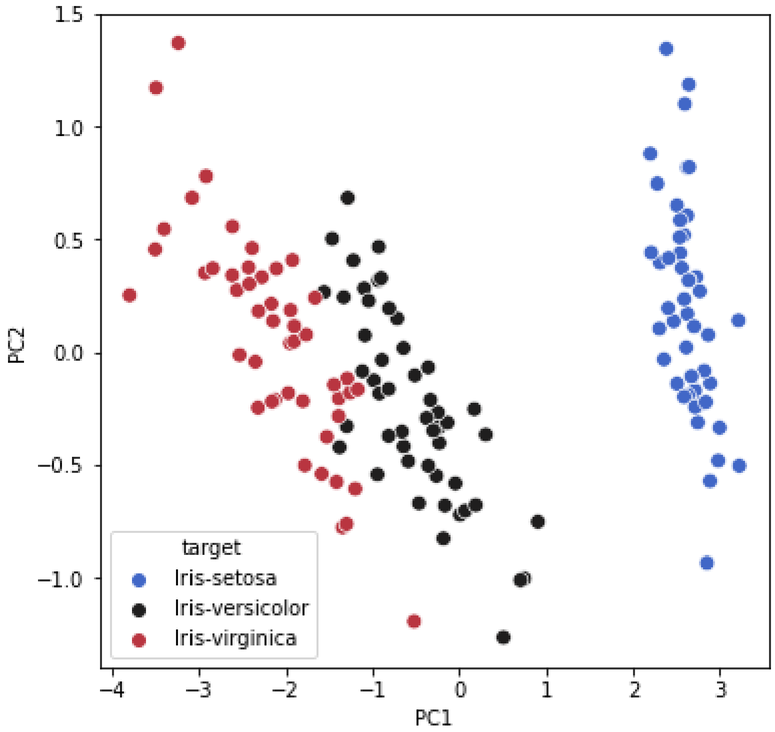
In [12]:

```
import seaborn as sb
import matplotlib.pyplot as plt

plt.figure(figsize = (6,6))
sb.scatterplot(data = principal_df , x = 'PC1',y = 'PC2' , hue = 'target' , s = 60 , pal
```

Out[12]:

```
&lt;AxesSubplot:xlabel='PC1', ylabel='PC2'&gt;
```



In [14]:

```
print(principal_df)
```

```
          PC1       PC2           target
0    2.684207  0.326607      Iris-setosa
1    2.715391 -0.169557      Iris-setosa
2    2.889820 -0.137346      Iris-setosa
3    2.746437 -0.311124      Iris-setosa
4    2.728593  0.333925      Iris-setosa
..        ...       ...              ...
145 -1.944017  0.187415   Iris-virginica
146 -1.525664 -0.375021   Iris-virginica
147 -1.764046  0.078519   Iris-virginica
148 -1.901629  0.115877   Iris-virginica
149 -1.389666 -0.282887   Iris-virginica

[150 rows x 3 columns]
```

In [ ]:

```
```